|  |  |
| --- | --- |
| Immagine che contiene testo, screenshot, clipart  Descrizione generata automaticamente | Immagine che contiene testo, segnale, clipart  Descrizione generata automaticamente |

Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

SISTEMA DI GESTIONE DI TRASPORTO FERROVIARIO

0252789

Simone Bucci

Indice

[1. Descrizione del Minimondo 2](file:///C:\Users\simon\Documents\Uni\Basi%20di%20Dati\progetto%20old\progetto-bd-simonebucci%20v1.0.docx#_Toc606296459)

[2. Analisi dei Requisiti 3](file:///C:\Users\simon\Documents\Uni\Basi%20di%20Dati\progetto%20old\progetto-bd-simonebucci%20v1.0.docx#_Toc1289394997)

[3. Progettazione concettuale 7](file:///C:\Users\simon\Documents\Uni\Basi%20di%20Dati\progetto%20old\progetto-bd-simonebucci%20v1.0.docx#_Toc2081466291)

[4. Progettazione logica 13](file:///C:\Users\simon\Documents\Uni\Basi%20di%20Dati\progetto%20old\progetto-bd-simonebucci%20v1.0.docx#_Toc2147004904)

[5. Progettazione fisica 19](file:///C:\Users\simon\Documents\Uni\Basi%20di%20Dati\progetto%20old\progetto-bd-simonebucci%20v1.0.docx#_Toc518560220)

[Appendice: Implementazione 35](file:///C:\Users\simon\Documents\Uni\Basi%20di%20Dati\progetto%20old\progetto-bd-simonebucci%20v1.0.docx#_Toc403811585)

# Descrizione del Minimondo

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | Si vuole realizzare un sistema informativo per la gestione dell’operatività di un’azienda di trasporto pubblico ferroviario ad alta velocità. I treni gestiti dal servizio sono caratterizzati da una matricola (codice univoco numerico di quattro cifre). Ogni veicolo è anche associato ad una data di acquisto e ad uno storico di manutenzione. È inoltre di interesse mantenere informazioni legate alla marca e modello delle locomotrici e vagoni, tenendo traccia anche dei treni adibiti al trasporto merci o passeggeri. Di ciascun treno è di interesse conoscere anche da quanti vagoni questo è composto. Per i treni adibiti al trasporto passeggeri sono memorizzati il numero di carrozze di prima e seconda classe, il numero massimo di passeggeri che possono viaggiare in ciascun vagone (con riferimento ai singoli posti). I vagoni di un treno merci devono essere caratterizzati dalla portata massima (in tonnellate).  Ciascuna tratta ha un insieme di fermate identificate dal nome della stazione, dalla città e dalla provincia in cui si trova. Inoltre, per ciascuna tratta, vengono mantenuti i capilinea di partenza ed arrivo. Ciascuna fermata è associata all’orario di arrivo e partenza previsti, eccezion fatta per i capilinea in cui uno solo dei due orari è disponibile. Ogni tratta viene coperta da un numero predefinito di treni, la cui associazione viene gestita dai gestori del servizio. I gestori possono, su base periodica, modificare il numero di treni operanti su ciascuna tratta. Gli amministratori del servizio gestiscono anche i conducenti, identificati da un codice fiscale, un nome, un cognome, una data di nascita ed un luogo di nascita.  Nella gestione degli orari di lavoro, i gestori del servizio devono garantire che ciascun macchinista non effettui più di 5 turni settimanali, per un massimo di 4 ore di lavoro. A ciascun treno passeggeri deve anche essere associato un capotreno, soggetto agli stessi vincoli orari. La gestione dei turni avviene da parte dei gestori del servizio su base mensile. Qualora un conducente o un capotreno si ponga in malattia, i gestori del servizio devono poter indicare che il lavoratore non ha coperto il turno per malattia e identificare un nuovo lavoratore cui assegnare la sostituzione del turno.  Ogni lavoratore ha la possibilità di generare un report sui propri turni di lavoro, su base settimanale, riportante l’indicazione degli orari e dei treni in cui esso è coinvolto.  Si vuole anche realizzare, per quanto riguarda i treni passeggeri, una funzionalità di prenotazione di biglietto. All’atto di acquisto di un biglietto (identificato per tratta e cui viene associato un posto disponibile sul treno con l’aggiunta di un codice di prenotazione univoco) l’acquirente deve indicare il proprio nome, cognome data di nascita, codice fiscale e numero di carta di credito.  I gestori del servizio devono poter associare a ciascun vagone di un treno merci la tipologia della merce trasportata, della massa complessiva, della tratta su cui tale merce viene trasportata. Si vuole inoltre tenere traccia delle informazioni di fatturazione delle aziende che inviano e ricevono la merce. Per semplicità, si può assumere che in un vagone siano trasportate merci di una sola società, dirette ad una sola società.  I controllori devono poter verificare la validità di un biglietto, partendo dal codice di prenotazione. I controllori hanno accesso a tutte le informazioni del passeggero e possono contrassegnare la prenotazione come “valida ed utilizzata”.  Gli addetti alla manutenzione possono inserire un report di manutenzione, indicante (in testo libero) quali riparazioni sono state effettuate in quale data. Queste informazioni sono associate alla singola locomotrice o al singolo vagone. |

# Analisi dei Requisiti

## Identificazione dei termini ambigui e correzioni possibili

|  |  |  |  |
| --- | --- | --- | --- |
| Linea | Termine | Nuovo termine | Motivo correzione |
| 3 | Veicolo | Treno | Fanno riferimento allo stesso oggetto, il treno. |
| 8 | Carrozze | Vagoni | Fanno riferimento allo stesso oggetto, il vagone. |
| 11 | Tratta | Servizio Ferroviario | La tratta identifica il percorso (da stazione capolinea a stazione di arrivo) il servizio ferroviario copre una tratta in uno specifico giorno ed a specifici orari (orario arrivo e partenza per ciascuna fermata). |
| 15 | Gestori | Amministratori | Fanno riferimento alla stessa persona, l’amministratore di sistema. |
| 16 | Gestori | Amministratori | Fanno riferimento alla stessa persona, l’amministratore di sistema. |
| 19 | Gestori | Amministratori | Fanno riferimento alla stessa persona, l’amministratore di sistema. |
| 20 | Macchinista | Conducente | Fanno riferimento alla stessa persona, il conducente. |
| 22 | Gestori | Amministratori | Fanno riferimento alla stessa persona, l’amministratore di sistema. |
| 23 | Gestori | Amministratori | Fanno riferimento alla stessa persona, l’amministratore di sistema. |
| 31 | Acquirente | Passeggero | Fanno riferimento alla stessa persona, il passeggero. |
| 33 | Gestori | Amministratori | Fanno riferimento alla stessa persona, l’amministratore di sistema. |
| 37 | Società | Azienda | Fanno riferimento alla stessa entità, l’azienda. |

### Specifica disambiguata

|  |
| --- |
| Si vuole realizzare un sistema informativo per la gestione dell’operatività di un’azienda di trasporto pubblico ferroviario ad alta velocità. I treni gestiti dal servizio sono caratterizzati da una matricola (codice univoco numerico di quattro cifre). Ogni treno è anche associato ad una data di acquisto e ad uno storico di manutenzione. È inoltre di interesse mantenere informazioni legate alla marca e modello delle locomotrici e vagoni, tenendo traccia anche dei treni adibiti al trasporto merci o passeggeri. Di ciascun treno è di interesse conoscere anche da quanti vagoni questo è composto. Per i treni adibiti al trasporto passeggeri sono memorizzati il numero di vagoni di prima e seconda classe, il numero massimo di passeggeri che possono viaggiare in ciascun vagone (con riferimento ai singoli posti). I vagoni di un treno merci devono essere caratterizzati dalla portata massima (in tonnellate).  Ciascun servizio ferroviario copre una tratta e ha un insieme di fermate identificate dal nome della stazione, dalla città e dalla provincia in cui si trova. Inoltre, per ciascuna tratta, vengono mantenuti i capilinea di partenza ed arrivo. Ciascuna fermata è associata all’orario di arrivo e partenza previsti, eccezion fatta per i capilinea in cui uno solo dei due orari è disponibile. Ogni tratta viene coperta da un numero predefinito di treni, la cui associazione viene gestita dagli amministratori del servizio. Gli amministratori possono, su base periodica, modificare il numero di treni operanti su ciascuna tratta. Gli amministratori del servizio gestiscono anche i conducenti, identificati da un codice fiscale, un nome, un cognome, una data di nascita ed un luogo di nascita.  Nella gestione degli orari di lavoro, gli amministratori del servizio devono garantire che ciascun conducente non effettui più di 5 turni settimanali, per un massimo di 4 ore di lavoro. A ciascun treno passeggeri deve anche essere associato un capotreno, soggetto agli stessi vincoli orari. La gestione dei turni avviene da parte degli amministratori del servizio su base mensile. Qualora un conducente o un capotreno si ponga in malattia, gli amministratori del servizio devono poter indicare che il lavoratore non ha coperto il turno per malattia e identificare un nuovo lavoratore cui assegnare la sostituzione del turno.  Ogni lavoratore ha la possibilità di generare un report sui propri turni di lavoro, su base settimanale, riportante l’indicazione degli orari e dei treni in cui esso è coinvolto.  Si vuole anche realizzare, per quanto riguarda i treni passeggeri, una funzionalità di prenotazione di biglietto. All’atto di acquisto di un biglietto (identificato per tratta e cui viene associato un posto disponibile sul treno con l’aggiunta di un codice di prenotazione univoco) il passeggero deve indicare il proprio nome, cognome, data di nascita, codice fiscale e numero di carta di credito.  I gestori del servizio devono poter associare a ciascun vagone di un treno merci la tipologia della merce trasportata, della massa complessiva, della tratta su cui tale merce viene trasportata. Si vuole inoltre tenere traccia delle informazioni di fatturazione delle aziende che inviano e ricevono la merce. Per semplicità, si può assumere che in un vagone siano trasportate merci di una sola azienda, dirette ad una sola azienda.  I controllori devono poter verificare la validità di un biglietto, partendo dal codice di prenotazione. I controllori hanno accesso a tutte le informazioni del passeggero e possono contrassegnare la prenotazione come “valida ed utilizzata”.  Gli addetti alla manutenzione possono inserire un report di manutenzione, indicante (in testo libero) quali riparazioni sono state effettuate in quale data. Queste informazioni sono associate alla singola locomotrice o al singolo vagone. |

## Glossario dei Termini

|  |  |  |  |
| --- | --- | --- | --- |
| Termine | Descrizione | Sinonimi | Collegamenti |
| Treno | Composto da una locomotiva e uno o più vagoni. | Veicolo | Vagoni, Locomotiva |
| Vagone | Componente di un treno | Carrozza | Treno |
| Locomotiva | Componente di un treno |  | Treno |
| Tratte | Ogni tratta può essere percorsa da uno più treni e può comprendere diverse fermate e due capilinea. |  | Treno, Fermata |
| Fermata | Stazione in cui si fermano i treni in una tratta |  | Tratta |
| Amministratore | Ente che gestisce il sistema | Gestore |  |
| Conducente | Lavoratore a cui vengono assegnati dei turni per condurre un treno. | Macchinista | Treno |
| Controllore | Lavoratore che effettua la verifica e validazione del biglietto passeggero |  |  |
| Capotreno | Lavoratore a cui vengono assegnati dei turni per un treno passeggeri. |  | Treno |
| Manutentore | Lavoratore che esegue attività di manutenzione sui treni e che vengono registrate sullo storico manutenzioni |  |  |
| Lavoratore | Impiegato della società di gestione trasporto pubblico ferroviaria |  | Conducente, Capotreno |
| Azienda | Ente che spedisce o riceve merce | Società |  |
| Acquirente | Persona che prenota un posto passeggero acquistando un biglietto | Passeggero |  |

## Raggruppamento dei requisiti in insiemi omogenei

|  |
| --- |
| Frasi relative a treni |
| I treni gestiti dal servizio sono caratterizzati da una matricola (codice univoco numerico di quattro cifre). Ogni treno è anche associato ad una data di acquisto e ad uno storico di manutenzione. È inoltre di interesse mantenere informazioni legate alla marca e modello delle locomotrici e vagoni, tenendo traccia anche dei treni adibiti al trasporto merci o passeggeri. Di ciascun treno è di interesse conoscere anche da quanti vagoni questo è composto. Per i treni adibiti al trasporto passeggeri sono memorizzati il numero di vagoni di prima e seconda classe, il numero massimo di passeggeri che possono viaggiare in ciascun vagone (con riferimento ai singoli posti). I vagoni di un treno merci devono essere caratterizzati dalla portata massima (in tonnellate).  Gli addetti alla manutenzione possono inserire un report di manutenzione, indicante (in testo libero) quali riparazioni sono state effettuate in quale data. Queste informazioni sono associate alla singola locomotrice o al singolo vagone. |

|  |
| --- |
| Frasi relative a tratte |
| Ciascun servizio ferroviario copre una tratta e ha un insieme di fermate identificate dal nome della stazione, dalla città e dalla provincia in cui si trova. Inoltre, per ciascuna tratta, vengono mantenuti i capilinea di partenza ed arrivo. Ciascuna fermata è associata all’orario di arrivo e partenza previsti, eccezion fatta per i capilinea in cui uno solo dei due orari è disponibile. Ogni tratta viene coperta da un numero predefinito di treni, la cui associazione viene gestita dagli amministratori del servizio. Gli amministratori possono, su base periodica, modificare il numero di treni operanti su ciascuna tratta. |

|  |
| --- |
| Frasi relative a lavoratori |
| Gli amministratori del servizio gestiscono anche i conducenti, identificati da un codice fiscale, un nome, un cognome, una data di nascita ed un luogo di nascita.  Nella gestione degli orari di lavoro, gli amministratori del servizio devono garantire che ciascun conducente non effettui più di 5 turni settimanali, per un massimo di 4 ore di lavoro. A ciascun treno passeggeri deve anche essere associato un capotreno, soggetto agli stessi vincoli orari. La gestione dei turni avviene da parte egli amministratori del servizio su base mensile. Qualora un conducente o un capotreno si ponga in malattia, gli amministratori del servizio devono poter indicare che il lavoratore non ha coperto il turno per malattia e identificare un nuovo lavoratore cui assegnare la sostituzione del turno.  Ogni lavoratore ha la possibilità di generare un report sui propri turni di lavoro, su base settimanale, riportante l’indicazione degli orari e dei treni in cui esso è coinvolto. |

|  |
| --- |
| Frasi relative a passeggeri |
| Si vuole anche realizzare, per quanto riguarda i treni passeggeri, una funzionalità di prenotazione di biglietto. All’atto di acquisto di un biglietto (identificato per tratta e cui viene associato un posto disponibile sul treno con l’aggiunta di un codice di prenotazione univoco) l’acquirente deve indicare il proprio nome, cognome data di nascita, codice fiscale e numero di carta di credito.  I controllori devono poter verificare la validità di un biglietto, partendo dal codice di prenotazione. I controllori hanno accesso a tutte le informazioni del passeggero e possono contrassegnare la prenotazione come “valida ed utilizzata”. |

|  |
| --- |
| Frasi relative a merci |
| I gestori del servizio devono poter associare a ciascun vagone di un treno merci la tipologia della merce trasportata, della massa complessiva, della tratta su cui tale merce viene trasportata. Si vuole inoltre tenere traccia delle informazioni di fatturazione delle aziende che inviano e ricevono la merce. Per semplicità, si può assumere che in un vagone siano trasportate merci di una sola azienda, dirette ad una sola azienda. |

# Progettazione concettuale

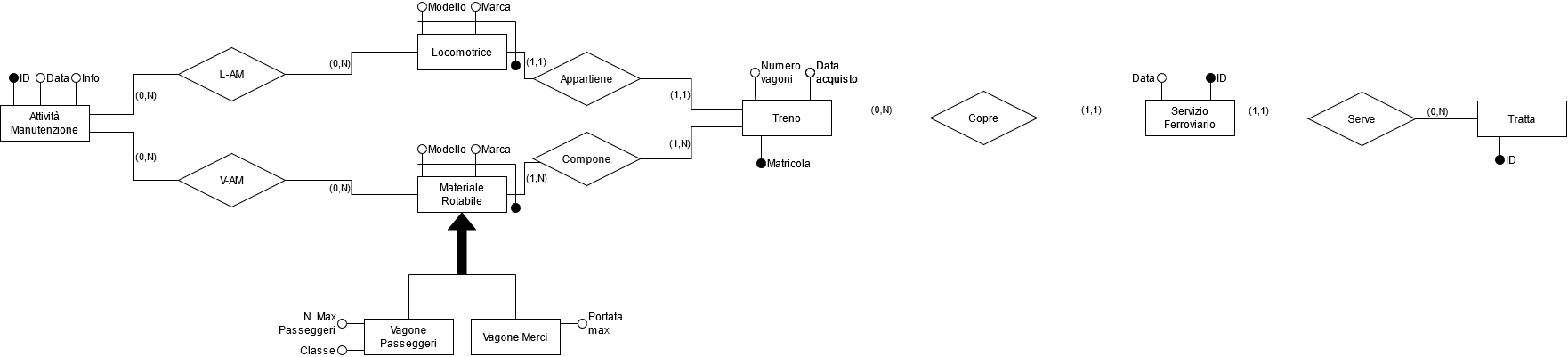
## Costruzione dello schema E-R

I concetti chiave sono il treno e la tratta che copre (il percorso tra due capolinea in una specifica ora del giorno). Per identificare l'occorrenza della copertura di una tratta nei diversi giorni dell'anno è stata aggiunta l’entità “Servizio Ferroviario” che potrà essere oggetto di prenotazione da parte dei passeggeri (servizio passeggeri) o da parte delle aziende (servizio merci).

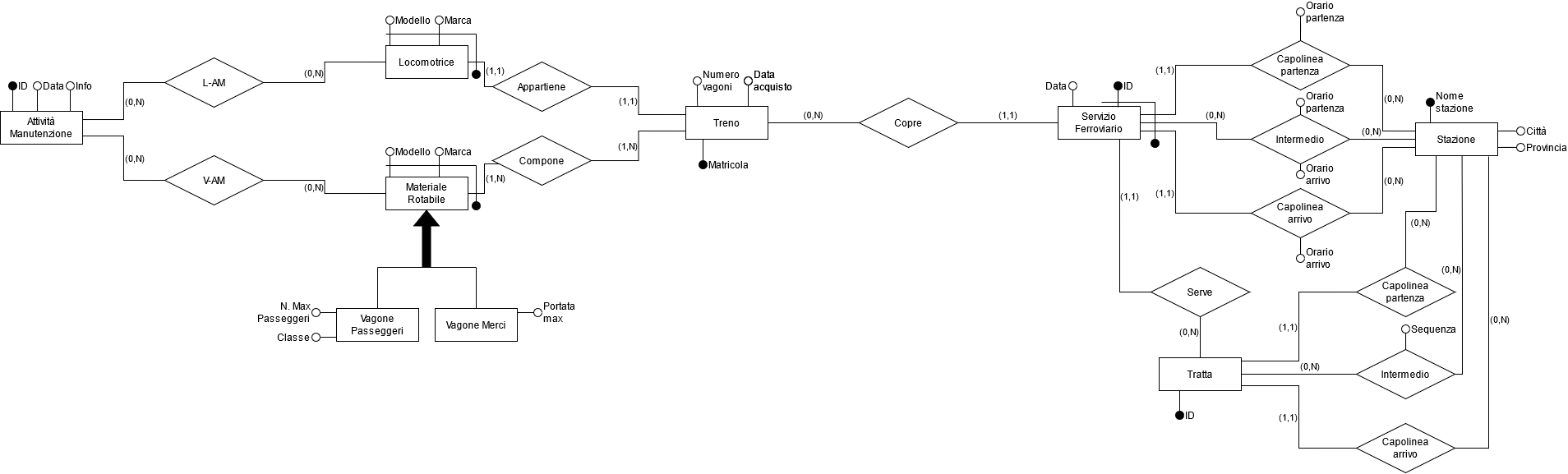


Il treno è composto da una locomotrice e più vagoni che possono essere di tipo passeggeri o merci. Per questo vengono create due entità “Locomotrice” e “Materiale Rotabile”, in particolare per il materiale rotabile viene adottata una generalizzazione per differenziare i due tipi di vagone “Passeggeri” e “Merci”.

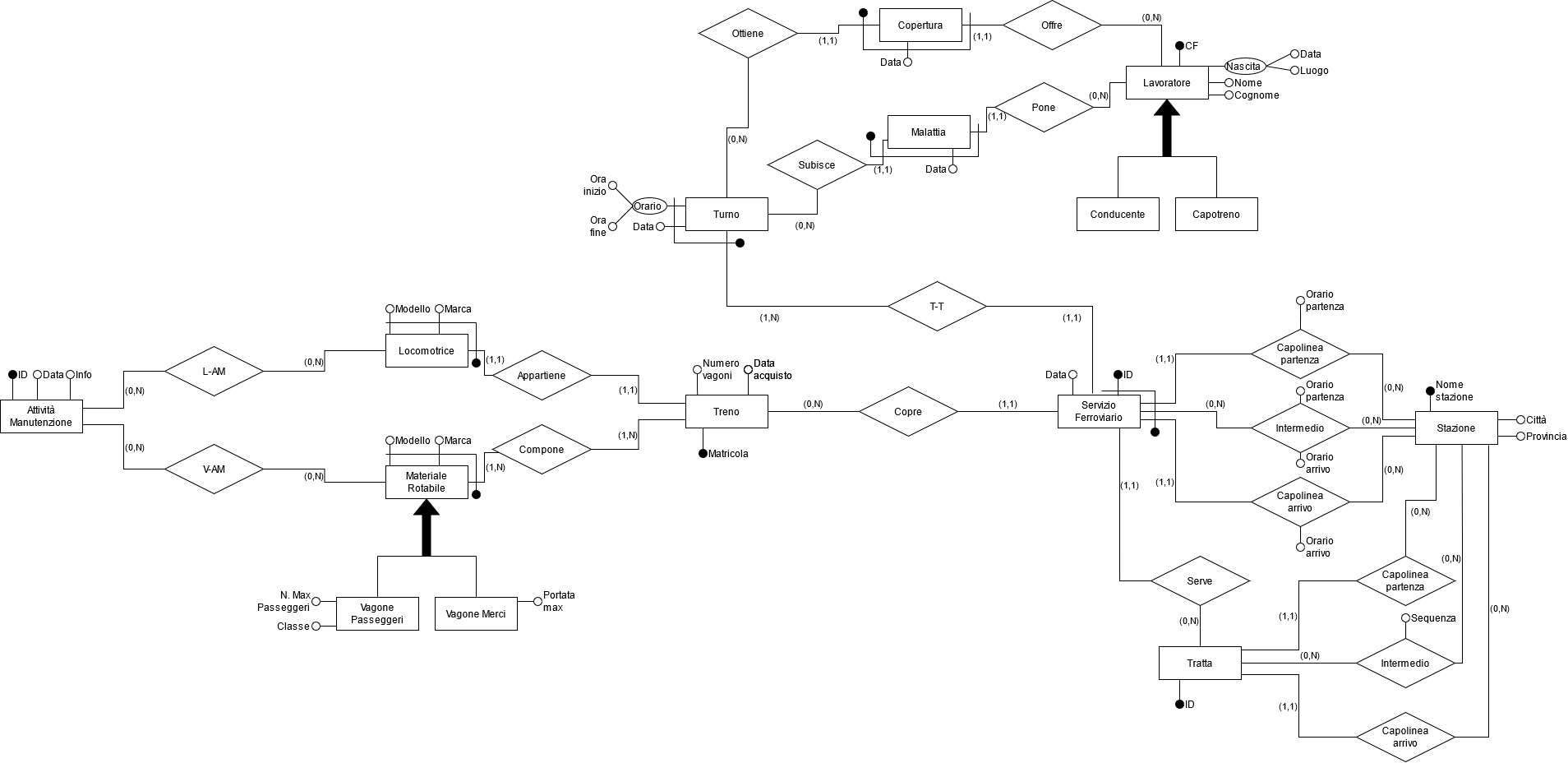
Viene aggiunta inoltre l’entità “Attività Manutenzione” per lo storico degli interventi effettuati sulle singole locomotrici e vagoni.



Ogni tratta è composta da un capolinea di partenza, uno di arrivo e da alcune fermate intermedie. Viene quindi creata una entità “Fermata” che verrà messa in relazione con “Tratta” e “Servizio Ferroviario” (per ottenere anche gli orari effettivi del viaggio) a seconda che sia uno dei due capolinea o una fermata intermedia.

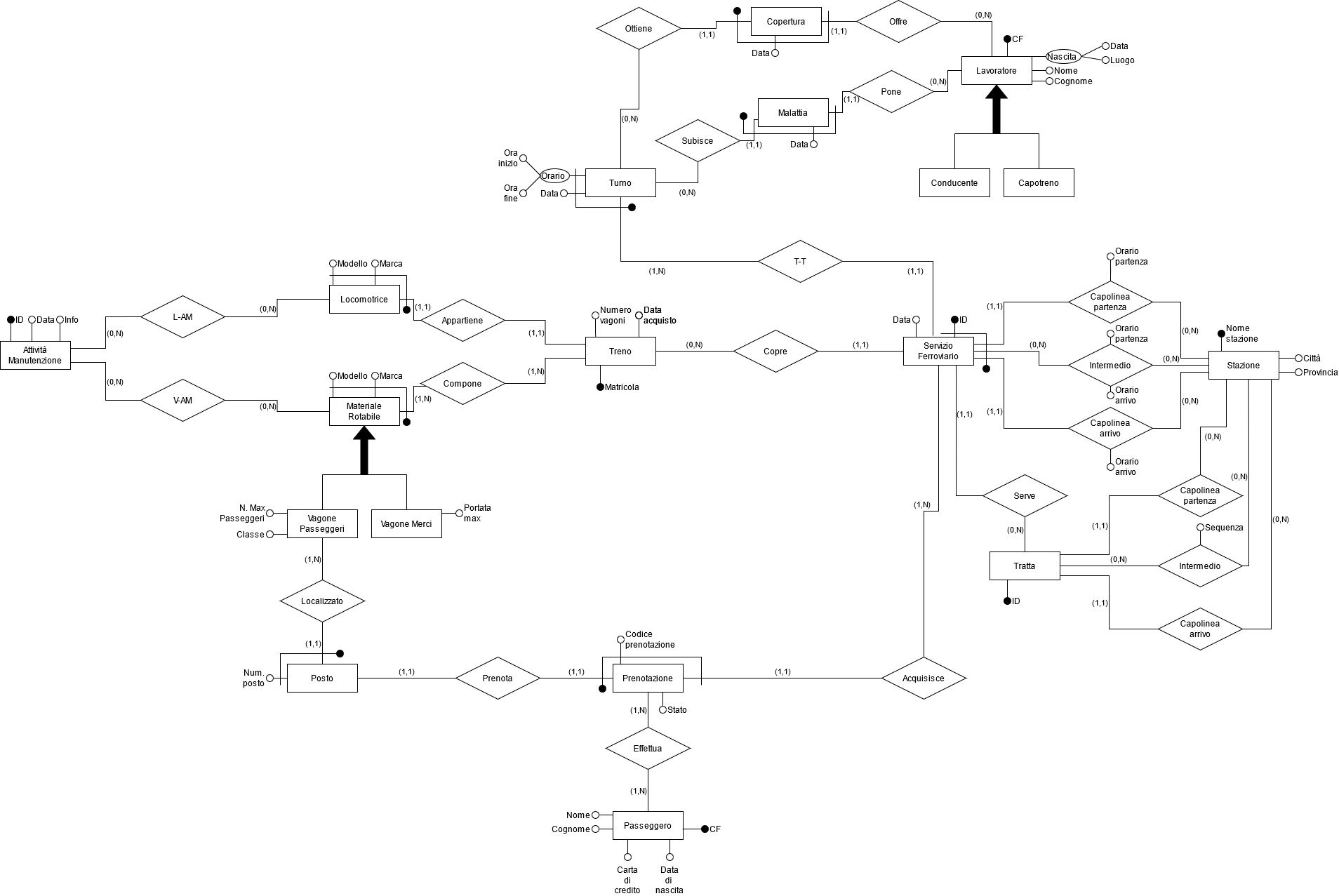


I lavoratori possono coprire o dichiararsi malati per determinati turni su determinati treni che coprono delle tratte. Sono state introdotte, quindi, due entità “Copertura” e “Malattia” messe in relazione con il lavoratore e con il turno. Ogni turno è in relazione con una o più tratte.



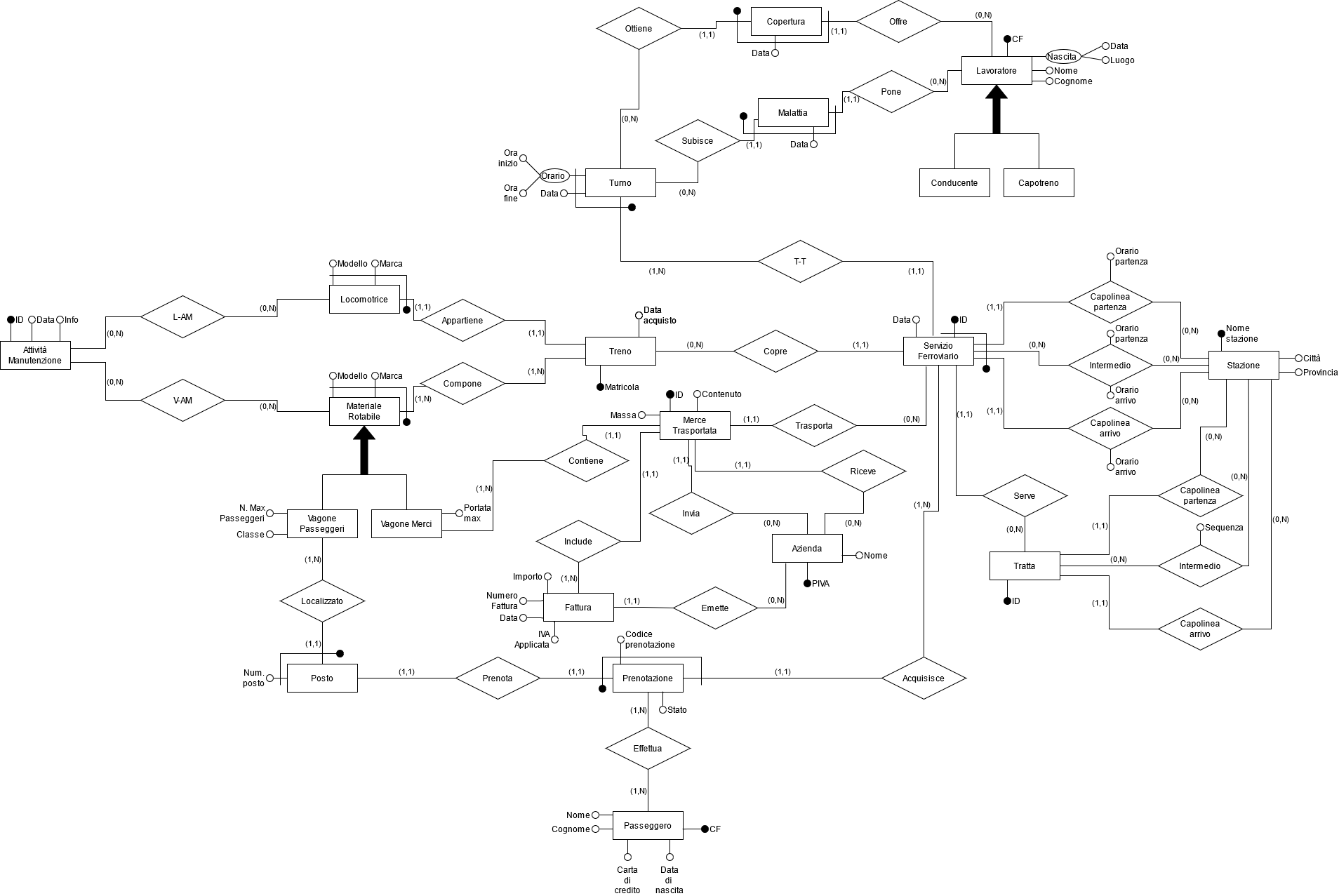
I passeggeri possono prenotare un posto su un vagone passeggeri di un treno che copre una determinata tratta. È stata creata quindi l’entità “Passeggero” che messa in relazione con “Prenotazione” indica la prenotazione di un determinato “Posto” su un determinato “Servizio Ferroviario” con le relative relazioni.

Le specifiche funzionali presuppongono, per semplicità, che un passeggero possa prenotare solo l’intera tratta e non percorsi parziali tra le fermate che compongono una tratta.



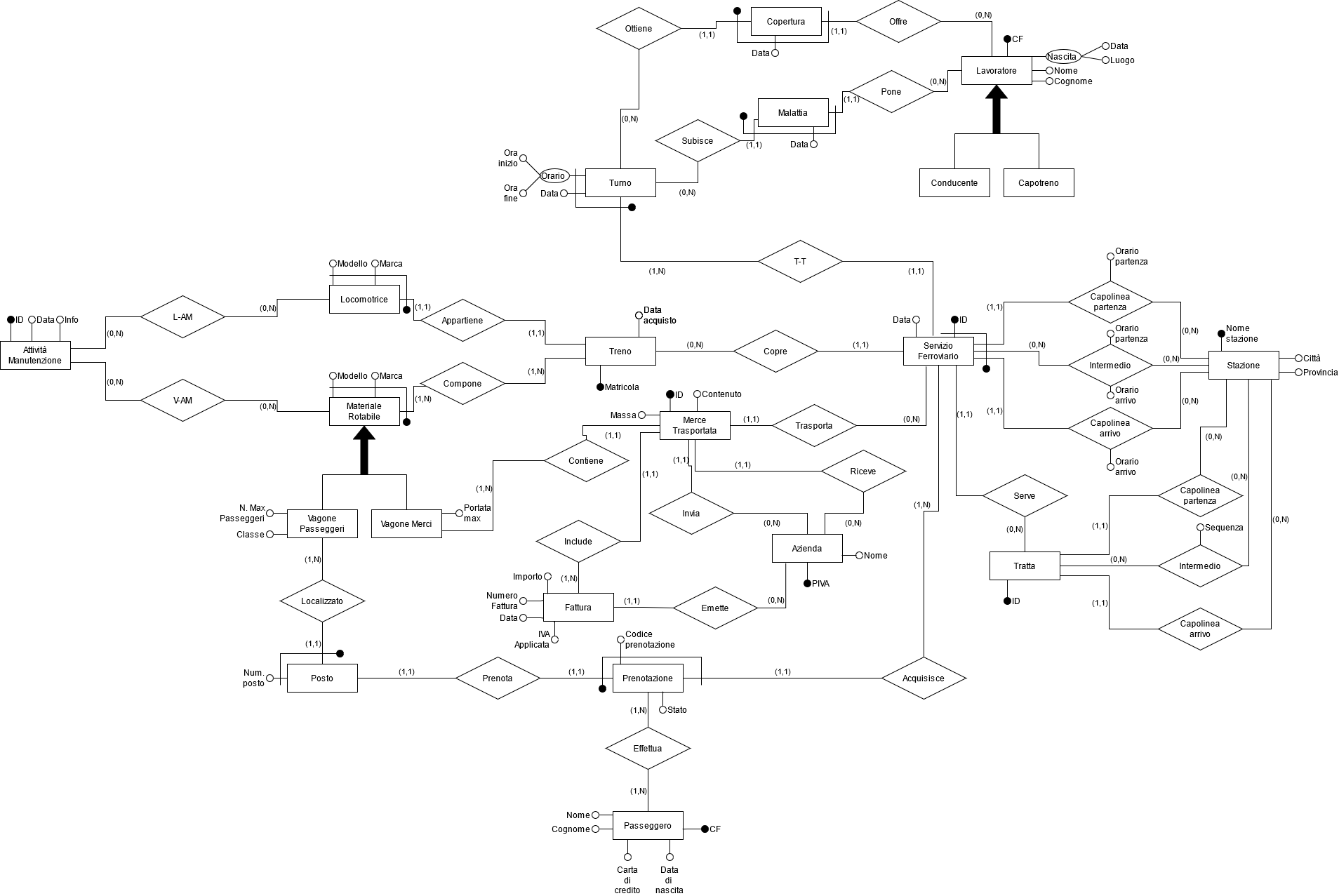
Le aziende possono far trasportare e ricevere merci attraverso i treni adibiti con vagoni merci su tratte specifiche. È stata creata infatti l’entità “Merce Trasportata” messa in relazione con il vagone merci e il servizio ferroviario su cui viaggia.

L'entità Merce Trasportata è inoltre in relazione con l'entità Fattura che riporta, oltre alle informazioni di fatturazione (data, importo, IVA applicata), anche le informazioni della ditta mittente e della ditta destinataria (tramite le due relazioni esistenti tra Fattura e l'entità Azienda).



### Integrazione finale

Non sono stati trovati conflitti strutturali o sui nomi.



## Regole aziendali

* I turni settimanali di ciascun conducente e capotreno non devono essere maggiori di 5
* Il numero dei turni settimanali si ottengono dal numero di “Copertura” con cui il lavoratore è in relazione
* Le ore di lavoro di ciascun conducente e capotreno devono essere massimo 4
* Le ore di lavoro di ciascun lavoratore si ottengono sottraendo l’ora di fine da quella di inizio di ogni turno e sommandole tra loro nelle date di una stessa settimana
* Il numero di vagoni di prima e di seconda classe si ottiene contando le occorrenze degli attributi corrispondenti sulle istanze dell’entità “Vagone Passeggeri”
* La disponibilità di un posto si verifica cercando una istanza di prenotazione con il numero di posto sulla tratta di interesse.

## Dizionario dei dati

|  |  |  |  |
| --- | --- | --- | --- |
| Entità | Descrizione | Attributi | Identificatori |
| Treno | Veicolo composto da locomotrice e vagoni | Matricola, Data di acquisto | Matricola |
| Locomotrice | Componente trainante del treno | Modello, Marca | Modello, Marca, Treno |
| Materiale Rotabile | Componente che contiene passeggeri o merce del treno | Modello, Marca | Modello, Marca, Treno |
| Attività Manutenzione | Informazioni sull’avvenuta manutenzione di una locomotrice o vagone | ID, Data, Info | ID |
| Vagone Passeggeri | Tipologia di vagone | N.MaxPass. , Classe | Materiale Rotabile, Treno |
| Vagone Merci | Tipologia di vagone | Portata max | Materiale Rotabile, Treno |
| Posto | Posto di un vagone passeggeri | N. Posto | N. Posto, Materiale Rotabile, Treno |
| Prenotazione | Prenotazione di un posto | Codice prenotazione, Stato | Codice Prenotazione, Posto, Servizio Ferroviario |
| Passeggero | Persona che viaggia su un treno | Codice Fiscale, Nome, Cognome, Carta di credito, Data di nascita | Codice Fiscale |
| Merce Trasportata | Contenuto del vagone merci | ID, Contenuto, Massa | ID |
| Azienda | Azienda che usufruisce del trasporto per ricevere o inviare merci | Nome | PIVA |
| Servizio Ferroviario | Servizio di un treno che copre una tratta | ID | ID, Capolinea Partenza, Capolinea Arrivo, Intermedio, Fermata |
| Tratta | Percorso coperto da più treni che comprendono più fermate | ID | ID |
| Stazione | Stazione in cui i treni si possono fermare | Nome, Città, Provincia | Nome |
| Turno | Turno che svolge un lavoratore su un treno | Ora Inizio, Ora Fine, Data | Ora Inizio, Ora Fine, Data, Servizio Ferroviario |
| Copertura | Turno che un lavoratore copre in una determinata data | Data | Lavoratore, Turno, Data |
| Malattia | Turno in cui un lavoratore si dichiara in malattia in una determinata data | Data | Lavoratore, Turno, Data |
| Lavoratore | Persona che lavora per la società di gestione ferroviaria | CF, Data di nascita, Luogo di nascita, Nome, Cognome | CF |
| Fattura | Fattura emessa dall’azienda mittente della merce trasportata | Numero fattura, IVA applicata, Importo, Data | Numero fattura, Azienda |

# Progettazione logica

## Volume dei dati

|  |  |  |
| --- | --- | --- |
| Concetto nello schema | Tipo**[[1]](#footnote-1)** | Volume atteso |
| Treno | E | 500 |
| Locomotrice | E | 600 |
| Materiale Rotabile | E | 10000 |
| Vagone Passeggeri | E | 2000 |
| Vagone Merci | E | 8000 |
| Attività Manutenzione | E | 12000 |
| L-AM | R | 2000 |
| V-AM | R | 10000 |
| Posto | E | 100000 |
| Prenotazione | E | 1000000 |
| Effettua | R | 1000000 |
| Passeggero | E | 500000 |
| Merce Trasportata | E | 2000000 |
| Azienda | E | 10000 |
| Servizio Ferroviario | E | 4000 |
| Tratta | E | 100 |
| Intermedio | R | 16000 |
| Capolinea Partenza | R | 80 |
| Intermedio | R | 200 |
| Capolinea Arrivo | R | 80 |
| Stazione | E | 200 |
| Turno | E | 1000 |
| Copertura | E | 2000 |
| Malattia | E | 200 |
| Lavoratore | E | 3000 |
| Conducente | E | 1500 |
| Capotreno | E | 1500 |

Le relazioni con cardinalità (1,1) non vengono riportate poiché hanno cardinalità uguale alla entità referenziata.

I volumi attesi riportati ipotizzano una retention di un anno.

## Tavola delle operazioni

|  |  |  |
| --- | --- | --- |
| Cod. | Descrizione | Frequenza attesa |
| 1 | Assegnazione turni | 1/mese |
| 2 | Modifica numero treni su una tratta | 4/mese |
| 3 | Dichiarazione malattia | 16/mese |
| 4 | Sostituzione lavoratore in malattia | 16/mese |
| 5 | Aggiunta treno passeggero | 1/anno |
| 6 | Aggiunta tratta | 1/anno |
| 7 | Report lavoratore | 4/mese |
| 8 | Prenotazione biglietto | 80000/mese |
| 9 | Report manutenzione | 10/mese |
| 10 | Verifica biglietto | 80000/mese |
| 11 | Inserisci merce su un treno | 160000/mese |
| 12 | Registrazione azienda | 500/mese |
| 13 | Registrazione passeggero | 40000/mese |

## Costo delle operazioni

**Operazione 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Copertura | E | 1000 | S |

Costo = 2\*2000 = 4000

**Operazione 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Serve | R | 100 | S |

Costo = 2\*4000\*4/mese = 32000

**Operazione 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Malattia | E | 1 | S |

Costo = 2\*1\*16/mese = 32

**Operazione 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Offre | R | 3000 | L |
| Copertura | E | 1 | S |

Costo = (3000+2\*1)\*16/mese = 48032

**Operazione 5**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Treno | E | 1 | S |
| Locomotrice | E | 1 | S |
| Vagone | E | 25 | S |
| Posto | E | 30 | S |

Costo = (2\*1+2\*1+25\*2+30\*2)\*1/anno = 114

**Operazione 6**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Tratta | E | 1 | S |

Costo = (2\*1)\*1/anno = 2

**Operazione 7**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Turno | E | 1 | L |

Costo = (1)\*4/mese = 4

**Operazione 8**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Prenotazione | E | 1 | S |

Costo = (2\*1)\*80000/mese = 160000

**Operazione 9**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Attività Manutenzione | E | 1 | S |

Costo = (2\*1)\*10/mese = 20

**Operazione 10**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Prenotazione | E | 250000 | L |

Costo = (1000000)\*80000/mese = 80000000000

**Operazione 13**

|  |  |  |  |
| --- | --- | --- | --- |
| Concetto | Costrutto | Accessi | Tipo |
| Passeggero | E | 1 | S |

Costo = (2\*1)\* 40000/mese = 80000

## Ristrutturazione dello schema E-R

La generalizzazione che riguarda i due tipi di vagoni, passeggeri e merci è stata eliminata aggiungendo un attributo “Tipo” all’entità “Materiale Rotabile”, è stata inoltre accorpata l’entità “Locomotrice”. Quindi l'entità **Materiale Rotabile rappresenta tutte le tipologie di componenti che compongono un treno**: “Locomotore”, “Vagone Passeggeri”, “Vagone Merci” ed avranno quindi una gestione unificata dello storico manutenzioni.

La generalizzazione che riguarda i due tipi di lavoratori è stata eliminata aggiungendo un attributo “Tipo” in sostituzione delle due entità “Conducente” e “Capotreno”.

L’attributo “Numero vagoni” è stato eliminato per evitare ridondanze, in quanto determinabile dalla relazione che definisce la composizione del treno in uno o più vagoni.

L’attributo “Data” su “Copertura” e “Malattia” è stato eliminato per evitare ridondanze, in quanto determinabile dalla relazione con “Turno”.

È stato introdotta l'entità “Fermata” che unifica e risolve le relazioni tra “Servizio Ferroviario” e “Stazione” e tra “Tratta” e “Stazione”. Ogni tratta è quindi caratterizzata da più “Fermate” ciascuna identificata da un numero di sequenza della fermata nella tratta (oltre che ovviamente dagli orari di arrivo e partenza). La fermata con numero di sequenza 1 sarà quindi il capolinea di partenza, mentre quella con il numero di sequenza massimo sarà il capolinea di arrivo.

Per evitare ridondanze riguardo alle aziende che spediscono le merci con la relativa fattura sono state eliminate le relazioni tra “Azienda” e “Merce Trasportata”, collegando le aziende direttamente a “Fattura” in modo tale da registrare chi emette e chi riceve la fattura con cui viene trasportata una determinata merce.

## 

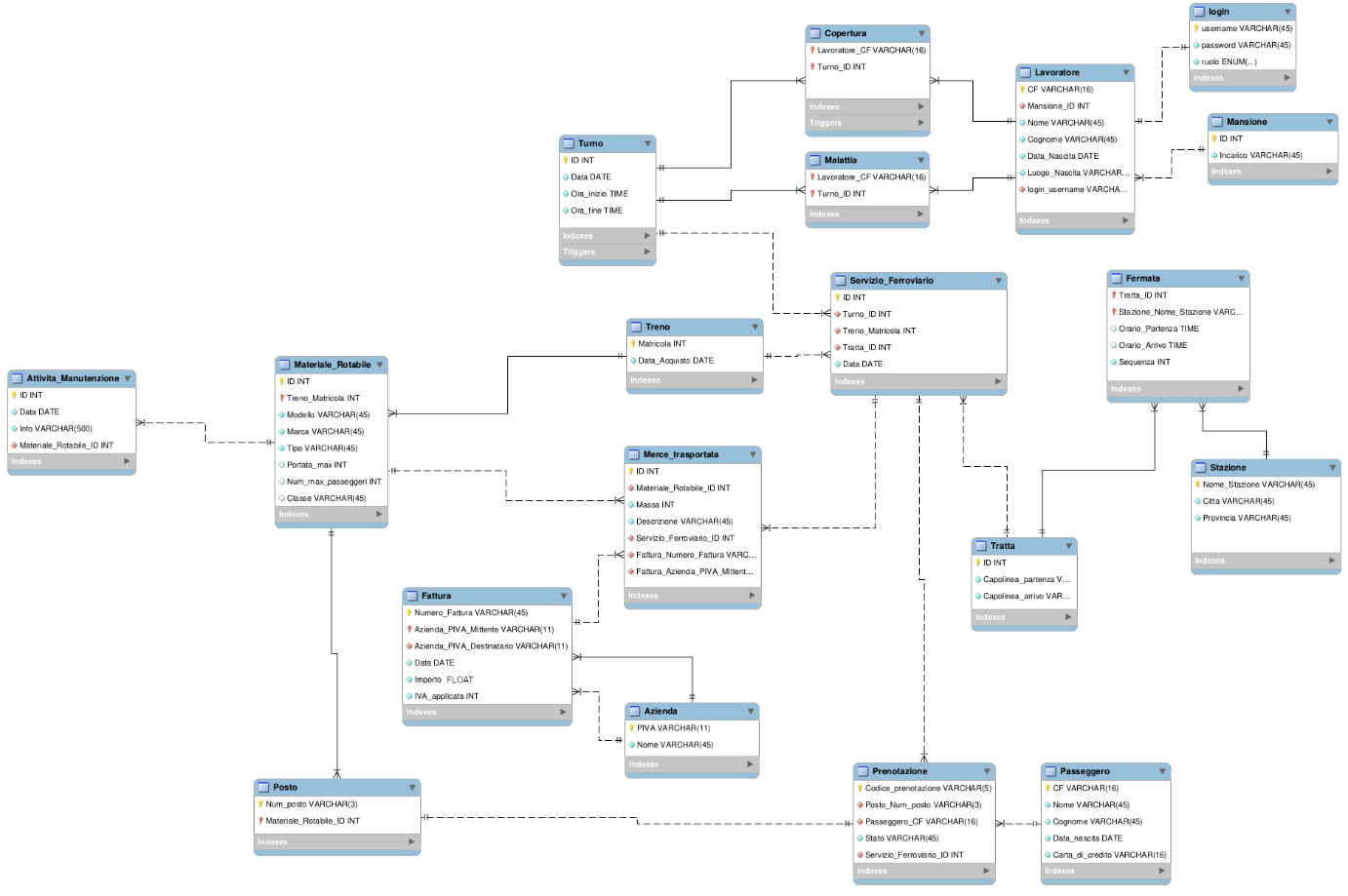
## Trasformazione di attributi e identificatori

Gli attributi riguardanti la nascita del lavoratore (data e luogo) vengono separati, eliminando l’attributo composto. Stessa cosa avviene per l’attributo composto orario nell’entità Turno che viene decomposto negli attributi orario inizio e orario fine.

Viene eliminato l’identificatore esterno su turno e su materiale rotabile inserendo un attributo ID ad entrambe le entità per facilitarne l’interazione.

## Traduzione di entità e associazioni

È stata inserita la tabella “login” per la gestione dei diversi utenti che potranno accedere alla base di dati con il ruolo assegnato. La tabella è identificata da un username e possiede inoltre un campo password ed un campo ruolo per indicare a quale ruolo (Amministratore, Lavoratore, Manutentore, Controllore) l’utente appartiene.



## Normalizzazione del modello relazionale

Lo schema si trova in 1NF perché tutti gli attributi sono definiti su un dominio indivisibile e non vi

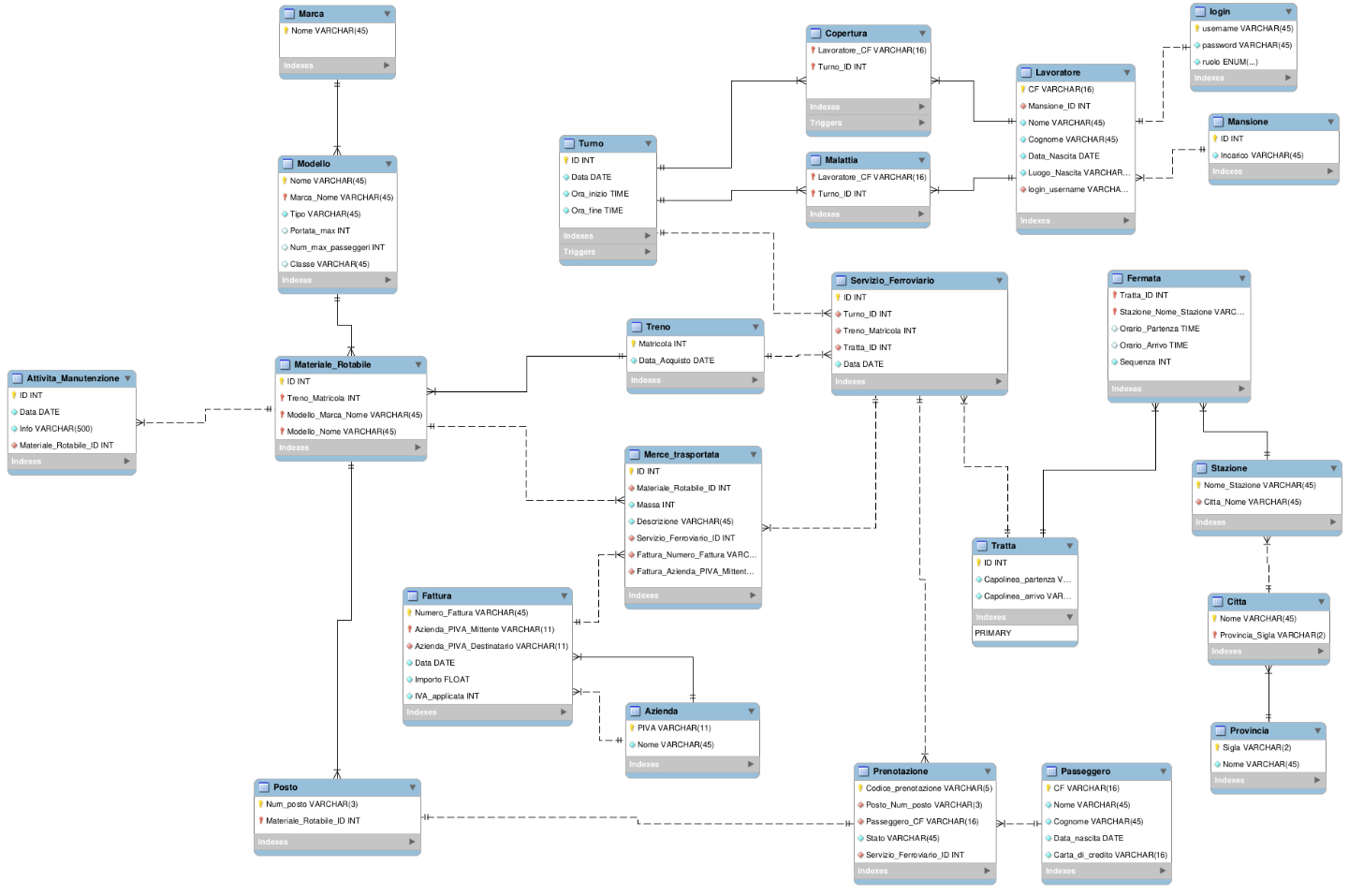
sono gruppi di attributi che si ripetono.

Lo schema si trova in 2NF perché tutti gli attributi non chiave dipendono dalla intera chiave.

Lo schema non si trova in 3NF perché tutti gli attributi non chiave non dipendono solamente dalla chiave. Infatti:

* L’attributo “Città” nella tabella “Stazione” è dipendente dall’attributo “Provincia”.
* L’attributo “Modello” nella tabella “Materiale Rotabile” è dipendente dall’attributo “Marca” così come gli attributi “Portata max”, “Num max passeggeri” e “Classe” sono dipendenti da “Modello”.

Per rispettare la 3NF sono state quindi create quattro ulteriori tabelle: “Città”, “Provincia”, “Modello” e “Marca” e le relative relazioni.



# Progettazione fisica

## Utenti e privilegi

I ruoli definiti e assegnati agli utenti ed i rispettivi privilegi sono:

* Amministratore: utente che gestisce la base di dati.
* Lavoratore: può accedere ai propri turni.
* Controllore: può verificare la validità di una prenotazione e contrassegnarla come utilizzata.
* Manutentore: può creare un resoconto di una attività di manutenzione.

A tale scopo segue l'elenco dei privilegi assegnati ai ruoli per ciascuna routine:

|  |  |  |
| --- | --- | --- |
| Permessi | | |
| Routine | Ruolo | Tipo |
| aggiungi\_lavoratore | Amministratore | EXECUTE |
| aggiungi\_azienda | Amministratore | EXECUTE |
| aggiungi\_passeggero | Amministratore | EXECUTE |
| aggiungi\_prenotazione | Amministratore | EXECUTE |
| aggiungi\_servizio | Amministratore | EXECUTE |
| aggiungi\_spedizione | Amministratore | EXECUTE |
| aggiungi\_turno | Amministratore | EXECUTE |
| lavoratori\_disponibili | Amministratore | EXECUTE |
| richiesta\_malattia | Amministratore | EXECUTE |
| assegna\_turno | Amministratore | EXECUTE |
| configurazione\_treno | Amministratore | EXECUTE |
| composizione\_treno | Amministratore | EXECUTE |
| modifica\_treno\_servizio | Amministratore | EXECUTE |
| modifica\_tratta\_servizio | Amministratore | EXECUTE |
| fermate\_tratta | Amministratore | EXECUTE |
| tratte\_giornaliere | Amministratore | EXECUTE |
| report\_lavoratore | Lavoratore | EXECUTE |
| verifica\_prenotazione | Controllore | EXECUTE |
| convalida\_prenotazione | Controllore | EXECUTE |
| report\_manutenzione | Manutentore | EXECUTE |

## Strutture di memorizzazione

|  |  |  |
| --- | --- | --- |
| Tabella Treno | | |
| Attributo | Tipo di dato | Attributi**[[2]](#footnote-2)** |
| Matricola | VARCHAR(4) | PK |
| Data\_Acquisto | DATE | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Materiale\_Rotabile | | |
| Attributo | Tipo di dato | Attributi**2** |
| ID | INT | PK |
| Treno\_Matricola | INT | PK |
| Modello\_Nome | VARCHAR(45) | PK |

|  |  |  |
| --- | --- | --- |
| Tabella Modello | | |
| Attributo | Tipo di dato | Attributi**2** |
| Nome | VARCHAR(45) | PK |
| Marca\_Nome | VARCHAR(45) | PK |
| Tipo | VARCHAR(45) | NN |
| Num\_max\_passeggeri | INT |  |
| Portata\_max | INT |  |
| Classe | VARCHAR(45) |  |

|  |  |  |
| --- | --- | --- |
| Tabella Marca | | |
| Attributo | Tipo di dato | Attributi2 |
| Nome | VARCHAR(45) | PK |

|  |  |  |
| --- | --- | --- |
| Tabella Attivita\_Manutenzione | | |
| Attributo | Tipo di dato | Attributi2 |
| ID | INT | PK, AI |
| Data | DATE | NN |
| Info | VARCHAR(45) | NN |
| Materiale\_Rotabile\_ID | INT | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Posto | | |
| Attributo | Tipo di dato | Attributi2 |
| Num\_posto | VARCHAR(3) | PK |
| Materiale\_Rotabile\_ID | INT | PK |

|  |  |  |
| --- | --- | --- |
| Tabella Prenotazione | | |
| Attributo | Tipo di dato | Attributi2 |
| Codice\_prenotazione | INT | PK |
| Posto\_Num\_posto | VARCHAR(3) | NN |
| Passeggero\_CF | VARCHAR(16) | NN |
| Servizio\_Ferroviario\_ID | INT | NN |
| Stato | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Passeggero | | |
| Attributo | Tipo di dato | Attributi2 |
| CF | VARCHAR(16) | PK |
| Nome | VARCHAR(45) | NN |
| Cognome | VARCHAR(45) | NN |
| Data\_nascita | DATE | NN |
| Carta\_di\_credito | VARCHAR(16) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Merce\_trasportata | | |
| Attributo | Tipo di dato | Attributi2 |
| ID | INT | PK |
| Servizio\_Ferroviario\_ID | INT | NN |
| Materiale\_Rotabile\_ID | INT | NN |
| Massa | INT | NN |
| Descrizione | VARCHAR(120) | NN |
| Azienda\_PIVA\_Mittente | VARCHAR(11) | NN |
| Fattura\_Numero\_Fattura | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Fattura | | |
| Attributo | Tipo di dato | Attributi2 |
| Numero\_Fattura | VARCHAR(45) | PK |
| Azienda\_PIVA\_Mittente | VARCHAR(11) | PK |
| Azienda\_PIVA\_Destinatario | VARCHAR(11) | NN |
| Data | DATE | NN |
| Importo | FLOAT | NN |
| IVA\_applicata | INT | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Azienda | | |
| Attributo | Tipo di dato | Attributi**2** |
| PIVA | VARCHAR(11) | PK |
| Nome | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Turno | | |
| Attributo | Tipo di dato | Attributi**2** |
| ID | INT | PK |
| Data | DATE | NN |
| Ora\_inizio | TIME | NN |
| Ora\_fine | TIME | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Copertura | | |
| Attributo | Tipo di dato | Attributi**2** |
| Lavoratore\_CF | VARCHAR(16) | PK |
| Turno\_ID | INT | PK |

|  |  |  |
| --- | --- | --- |
| Tabella Malattia | | |
| Attributo | Tipo di dato | Attributi**2** |
| Lavoratore\_CF | VARCHAR(16) | PK |
| Turno\_ID | INT | PK |

|  |  |  |
| --- | --- | --- |
| Tabella Lavoratore | | |
| Attributo | Tipo di dato | Attributi**2** |
| CF | VARCHAR(16) | PK |
| Mansione\_ID | INT | NN |
| Nome | VARCHAR(45) | NN |
| Cognome | VARCHAR(45) | NN |
| Data\_nascita | DATE | NN |
| Luogo\_Nascita | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Mansione | | |
| Attributo | Tipo di dato | Attributi**2** |
| ID | INT | PK |
| Incarico | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Servizio\_Ferroviario | | |
| Attributo | Tipo di dato | Attributi**2** |
| ID | INT | PK |
| Tratta\_ID | INT | NN |
| Treno\_Matricola | INT | NN |
| Turno\_ID | INT | NN |
| Data | DATE | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Tratta | | |
| Attributo | Tipo di dato | Attributi**2** |
| ID | INT | PK |
| Capolinea\_partenza | VARCHAR(45) | NN |
| Capolinea\_arrivo | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Stazione | | |
| Attributo | Tipo di dato | Attributi**2** |
| Nome\_Stazione | VARCHAR(45) | PK |
| Citta\_Nome | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Citta | | |
| Attributo | Tipo di dato | Attributi**2** |
| Nome | VARCHAR(45) | PK |
| Provincia\_sigla | VARCHAR(2) | PK |

|  |  |  |
| --- | --- | --- |
| Tabella Provincia | | |
| Attributo | Tipo di dato | Attributi**2** |
| Sigla | VARCHAR(2) | PK |
| Nome | VARCHAR(45) | NN |

|  |  |  |
| --- | --- | --- |
| Tabella Fermata | | |
| Attributo | Tipo di dato | Attributi**2** |
| Tratta\_ID | INT | PK |
| Stazione\_Nome\_Stazione | VARCHAR(45) | PK |
| Stazione\_Citta\_Nome | VARCHAR(45) | NN |
| Sequenza | INT | NN |
| Orario\_partenza | TIME |  |
| Orario\_arrivo | TIME |  |

|  |  |  |
| --- | --- | --- |
| Tabella login | | |
| Attributo | Tipo di dato | Attributi**2** |
| Username | VARCHAR(45) | PK |
| Password | VARCHAR(2) | NN |
| Ruolo | ENUM(‘amministratore’,’lavoratore’,’controllore’,’manutentore’) | NN |

## Indici

|  |  |
| --- | --- |
| Tabella Modello | |
| Indice Modello\_marca | Tipo**[[3]](#footnote-3)**: |
| fk\_Modello\_Marca1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Materiale\_Rotabile | |
| Indice Vagone | Tipo**3**: |
| fk\_Vagone\_Treno1\_idx | IDX |
| fk\_Materiale\_Rotabile\_Modello1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Attivita\_Manutenzione | |
| Indice Manutenzione | Tipo**3**: |
| fk\_Attivita\_Manutenzione\_Materiale\_Rotabile\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Posto | |
| Indice Posto | Tipo**3**: |
| fk\_Posto\_Materiale\_Rotabile1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Prenotazione | |
| Indice Prenotazione | Tipo**3**: |
| fk\_Prenotazione\_Posto1\_idx | IDX |
| fk\_Prenotazione\_Passeggero1\_idx | IDX |
| fk\_Prenotazione\_Servizio\_Ferroviario1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Merce\_Trasportata | |
| Indice Merce | Tipo**3**: |
| fk\_Merce\_trasportata\_Materiale\_Rotabile1\_idx | IDX |
| fk\_Merce\_trasportata\_Servizio\_Ferroviario1\_idx | IDX |
| fk\_Merce\_trasportata\_Azienda1\_idx | IDX |
| fk\_Merce\_trasportata\_Azienda2\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Copertura | |
| Indice Copertura | Tipo**3**: |
| fk\_Copertura\_Lavoratore1\_idx | IDX |
| fk\_Copertura\_Turno1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Malattia | |
| Indice Malattia | Tipo**3**: |
| fk\_Malattia\_Lavoratore1\_idx | IDX |
| fk\_Malattia\_Turno1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Lavoratore | |
| Indice Lavoratore | Tipo**3**: |
| fk\_Lavoratore\_Mansione1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Servizio\_Ferroviario | |
| Indice Servizio | Tipo**3**: |
| fk\_Tratta\_Turno1\_idx | IDX |
| fk\_Tratta\_Treno1\_idx | IDX |
| fk\_Itinerario\_Tratta1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Fermata | |
| Indice Fermata | Tipo**3**: |
| fk\_Fermata\_Stazione1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Stazione | |
| Indice Stazione | Tipo**3**: |
| fk\_Citta\_Citta1\_idx | IDX |

|  |  |
| --- | --- |
| Tabella Citta | |
| Indice Citta | Tipo**3**: |
| fk\_Citta\_Provincia1\_idx | IDX |

## Trigger

È stato implementato il trigger **Turno\_BEFORE\_INSERT** per controllare la correttezza dell’orario e per far sì che un turno non superi le 4 ore come da regola aziendale.

**CREATE** DEFINER **=** **CURRENT\_USER** **TRIGGER** `AziendaFerroviaria`**.**`Turno\_BEFORE\_INSERT` **BEFORE** **INSERT** **ON** `Turno` **FOR** **EACH** **ROW**

**BEGIN**

**IF(new.**Ora\_inizio **>** **new.**Ora\_fine**)** **THEN**

SIGNAL **SQLSTATE** '45000'

**SET** MESSAGE\_TEXT **=** 'Orari non validi'**;**

ELSEIF**(**TIMEDIFF**(new.**Ora\_fine**,** **new.**Ora\_inizio**)** **>** "04:00:00"**)** **THEN**

SIGNAL **SQLSTATE** '45000'

**SET** MESSAGE\_TEXT **=** 'Il turno deve essere di un massimo di 4 ore'**;**

**END** **IF;**

**END**

È stato implementato il trigger **Copertura\_BEFORE\_INSERT** per far sì che non vengano assegnati più di 5 turni alla settimana ad un lavoratore.

**CREATE** DEFINER **=** **CURRENT\_USER** **TRIGGER** `AziendaFerroviaria`**.**`Copertura\_BEFORE\_INSERT` **BEFORE** **INSERT** **ON** `Copertura` **FOR** **EACH** **ROW**

**BEGIN**

**DECLARE** n **INT;**

**DECLARE** d **DATE;**

**select** **data** **INTO** d **from** Turno **where** ID**=new.**Turno\_ID**;**

**SELECT** **count(\*)** **INTO** n **from** Copertura**,** Turno **where** Turno\_ID**=**ID **and** Lavoratore\_CF**=new.**Lavoratore\_CF **and** week**(**d**)=**week**(Data) and** **year(**d**)** **=** **year(Data);**

**IF(**n **>=** 5**)** **THEN**

SIGNAL **SQLSTATE** '45000'

**SET** MESSAGE\_TEXT **=** 'Massimo numero di turni raggiunto'**;**

**END** **IF;**

**END**

## Eventi

Non sono stati usati eventi nella applicazione.

## Viste

**Convoglio**

Vista che mostra tutti i treni con la loro composizione.

**CREATE** **VIEW** Convoglio **as** **SELECT** Treno\_Matricola**,** Modello\_Marca\_Nome**,** Modello\_Nome**,** Tipo**,** Portata\_max**,** Num\_max\_passeggeri**,**Classe

**FROM** Materiale\_Rotabile**,** Modello

**where** Modello\_Nome**=**Nome **and** Modello\_Marca\_Nome**=**Marca\_Nome

**order** **by** Treno\_Matricola**;**

**composizione\_treno**

Vista che mostra la composizione dei treni, con le varie specifiche dei vagoni e il loro numero.

**CREATE** **VIEW** `composizione\_treno` **AS**

**select** treno\_matricola**,** tipo**,** classe**,** **count(\*)** N\_Vagoni

**from** Convoglio

**where** **(**Classe **in** **(**'PRIMA'**,**'SECONDA'**)** **or** Tipo**=**'LOCOMOTORE' **or** Tipo**=**'MERCI'**)**

**group** **by** Treno\_Matricola**,** Tipo**,** Classe

**order** **by** Treno\_Matricola**;**

## Stored Procedures e transazioni

**aggiungi\_azienda**

Procedura che aggiunge un’azienda nella base di dati inserendone i dati nella tabella “Azienda”.

**CREATE** **PROCEDURE** `aggiungi\_azienda` **(IN** var\_piva **VARCHAR(**11**),** **IN** var\_nome **VARCHAR(**45**))**

**BEGIN**

**INSERT** **INTO** Azienda

**(**`PIVA`**,**

`Nome`**)**

**VALUES**

**(**var\_piva**,**

var\_nome**);**

**END**

**aggiungi\_lavoratore**

Procedura che aggiunge un lavoratore nella base di dati inserendone i dati nella tabella “Lavoratore” per i dettagli sul lavoratore e “login” per la gestione degli accessi.

**CREATE** **PROCEDURE** `aggiungi\_lavoratore` **(IN** var\_cf **VARCHAR(**16**),** **IN** var\_mansione **INT,** **IN** var\_nome **VARCHAR(**45**),** **IN** var\_cognome **VARCHAR(**45**),** **IN** var\_data\_nascita **DATE,** **IN** var\_luogo\_nascita **VARCHAR(**45**),** **IN** var\_username **VARCHAR(**45**),** **IN** var\_pass **VARCHAR(**45**))**

**BEGIN**

**start** **transaction;**

**INSERT** **INTO** login**(**`username`**,**`password`**,**`ruolo`**)**

**VALUES(**var\_username**,** var\_pass**,** '2'**);**

**INSERT** **INTO** Lavoratore**(**`CF`**,** `Mansione\_ID`**,** `Nome`**,** `Cognome`**,** `Data\_nascita`**,** `Luogo\_nascita`**,**`login\_username`**)**

**VALUES(**var\_cf**,** var\_mansione**,** var\_nome**,** var\_cognome**,** var\_data\_nascita**,** var\_luogo\_nascita**,** var\_username**);**

**commit;**

**END**

**aggiungi\_passeggero**

Procedura che aggiunge un passeggero nella base di dati inserendone i dati nella tabella “Passeggero”.

**CREATE** **PROCEDURE** `aggiungi\_passeggero` **(IN** var\_cf **VARCHAR(**16**),** **IN** var\_nome **VARCHAR(**45**),** **IN** var\_cognome **VARCHAR(**45**),** **IN** var\_data\_nascita **DATE,** **IN** var\_carta **VARCHAR(**16**))**

**BEGIN**

**start** **transaction;**

**INSERT** **INTO** `AziendaFerroviaria`**.**`Passeggero`

**(**`CF`**,**

`Nome`**,**

`Cognome`**,**

`Data\_nascita`**,**

`Carta\_di\_credito`**)**

**VALUES**

**(**var\_cf**,**

var\_nome**,**

var\_cognome**,**

var\_data\_nascita**,**

var\_carta**);**

**commit;**

**END**

**aggiungi\_prenotazione**

Procedura che aggiunge una prenotazione controllando la disponibilità del posto su quel servizio ferroviario e inserendo i dati dell’acquirente e del servizio prenotato nella tabella “Prenotazione” nel caso in cui sia libero.

**CREATE** **PROCEDURE** `aggiungi\_prenotazione` **(IN** var\_codice\_prenotazione **VARCHAR(**5**),** **IN** var\_posto\_num **VARCHAR(**4**),** **IN** var\_passeggero\_cf **VARCHAR(**16**),** **IN** var\_servizio\_id **INT)**

**BEGIN**

**DECLARE** **EXIT** HANDLER **FOR** **sqlexception**

**BEGIN**

**rollback;**

resignal**;**

**END;**

**SET** **TRANSACTION** **ISOLATION** **LEVEL** serializable**;**

**start** **transaction;**

**if** **exists** **(Select** **\*** **from** Prenotazione **where** Posto\_num\_posto **=** var\_posto\_num **and**

Servizio\_Ferroviario\_ID **=** var\_servizio\_id**)** **then**

SIGNAL **SQLSTATE** '45000'

**SET** MESSAGE\_TEXT **=** 'Il posto selezionato è occupato.'**;**

**end** **if;**

**INSERT** **IGNORE** **INTO** `AziendaFerroviaria`**.**`Prenotazione`

**(**`Codice\_prenotazione`**,**

`Posto\_Num\_posto`**,**

`Passeggero\_CF`**,**

`Stato`**,**

`Servizio\_Ferroviario\_ID`**)**

**VALUES**

**(**var\_codice\_prenotazione**,**

var\_posto\_num**,**

var\_passeggero\_cf**,**

'VALIDO'**,**

var\_servizio\_id**);**

**commit;**

**END**

**aggiungi\_servizio**

Procedura che aggiunge un servizio nella base di dati inserendone i dati nella tabella “Servizio Ferroviario”.

**CREATE** **PROCEDURE** `aggiungi\_servizio` **(IN** var\_id **INT,** **IN** var\_turno **INT,** **IN** var\_treno **VARCHAR(**4**),** **IN** var\_tratta **INT,** **IN** var\_data **DATE)**

**BEGIN**

**DECLARE** **EXIT** HANDLER **FOR** **sqlexception**

**BEGIN**

**rollback;**

resignal**;**

**END;**

**SET** **TRANSACTION** **ISOLATION** **LEVEL** repeatable **read;**

**start** **transaction;**

**INSERT** **INTO** `AziendaFerroviaria`**.**`Servizio\_Ferroviario`

**(**`ID`**,**

`Turno\_ID`**,**

`Treno\_Matricola`**,**

`Tratta\_ID`**,**

`**Data**`**)**

**VALUES**

**(**var\_id**,**

var\_turno**,**

var\_treno**,**

var\_tratta**,**

var\_data**);**

**commit;**

**END**

**aggiungi\_spedizione**

Procedura che aggiunge una spedizione nella base di dati inserendone i dati insieme alle informazioni del mittente e del destinatario nella tabella “Merce\_Trasportata” solo dopo aver verificato che il vagone sul servizio ferroviario specificato sia di tipo merci e non sia già occupato.

**CREATE** **PROCEDURE** `aggiungi\_spedizione` **(IN** var\_merce\_id **INT,** **IN** var\_mittente **VARCHAR(**11**),** **IN** var\_servizio **INT,** **IN** var\_vagone\_id **INT,** **IN** var\_massa **INT,** **IN** var\_descrizione **VARCHAR(**120**),** **IN** var\_fattura **VARCHAR(**45**))**

**BEGIN**

**DECLARE** **EXIT** HANDLER **FOR** **sqlexception**

**BEGIN**

**rollback;**

resignal**;**

**END;**

**SET** **TRANSACTION** **ISOLATION** **LEVEL** serializable**;**

**start** **transaction;**

**start** **transaction;**

**if** **exists** **(Select** **\*** **from** Merce\_trasportata **where** Materiale\_Rotabile\_ID **=** var\_vagone\_id **and**

Servizio\_Ferroviario\_ID **=** var\_servizio**)** **then**

SIGNAL **SQLSTATE** '45000'

**SET** MESSAGE\_TEXT **=** 'Il vagone selezionato è occupato.'**;**

**end** **if;**

**if** **exists** **(Select** **\*** **from** Materiale\_Rotabile**,**Modello **where** ID **=** var\_vagone\_id **and** Modello\_Nome **=** Nome **and** Tipo **!=** 'MERCI'**)** **then**

SIGNAL **SQLSTATE** '45000'

**SET** MESSAGE\_TEXT **=** 'Il tipo di vagone selezionato non è corretto.'**;**

**end** **if;**

**if** **exists** **(Select** **\*** **from** Materiale\_Rotabile**,**Modello **where** ID **=** var\_vagone\_id **and** Modello\_Nome **=** Nome **and** Portata\_max **<** var\_massa**)** **then**

SIGNAL **SQLSTATE** '45000'

**SET** MESSAGE\_TEXT **=** 'Il limite di massa per il vagone è stato superato.'**;**

**end** **if;**

**INSERT** **IGNORE** **INTO** `AziendaFerroviaria`**.**`Merce\_trasportata`

**(**`ID`**,**

`Servizio\_Ferroviario\_ID`**,**

`Materiale\_Rotabile\_ID`**,**

`Massa`**,**

`Descrizione`**,**

`Fattura\_Numero\_Fattura`**,**

`Fattura\_Azienda\_PIVA\_Mittente`**)**

**VALUES**

**(**var\_merce\_id**,**

var\_servizio**,**

var\_vagone\_id**,**

var\_massa**,**

var\_descrizione**,**

var\_fattura**,**

var\_mittente**);**

**commit;**

**END**

**aggiungi\_turno**

Procedura che aggiunge un turno nella base di dati inserendone i dati nella tabella “Turno”.

**CREATE** **PROCEDURE** `aggiungi\_turno` **(IN** var\_id **INT,** **IN** var\_data **DATE,** **IN** var\_inizio **TIME,** **IN** var\_fine **TIME)**

**BEGIN**

**DECLARE** **EXIT** HANDLER **FOR** **sqlexception**

**BEGIN**

**rollback;**

resignal**;**

**END;**

**start** **transaction;**

**INSERT** **INTO** `AziendaFerroviaria`**.**`Turno`

**(**`ID`**,**

`**Data**`**,**

`Ora\_inizio`**,**

`Ora\_fine`**)**

**VALUES**

**(**var\_id**,**

var\_data**,**

var\_inizio**,**

var\_fine**);**

**commit;**

**END**

**assegna\_turno**

Procedura che assegna ad un lavoratore un turno controllandone prima la disponibilità verificando che per quel turno non sia in malattia, inserisce i dati nella tabella “Copertura”.

**CREATE** **PROCEDURE** `assegna\_turno` **(IN** var\_lavoratore **VARCHAR(**16**),** **IN** var\_turno **INT)**

**BEGIN**

**DECLARE** **EXIT** HANDLER **FOR** **sqlexception**

**BEGIN**

**rollback;**

resignal**;**

**END;**

**set** **transaction** **isolation** **level** serializable**;**

**start** **transaction;**

**if** **exists** **(Select** **\*** **from** Malattia **where** Lavoratore\_CF **=** var\_lavoratore **and**

Turno\_ID **=** var\_turno**)** **then**

SIGNAL **SQLSTATE** '45000'

**SET** MESSAGE\_TEXT **=** 'Lavoratore in malattia'**;**

**end** **if;**

**INSERT** **IGNORE** **INTO** Copertura **VALUES** **(**var\_lavoratore**,** var\_turno**);**

**commit;**

**END**

**composizione\_treno**

Procedura che restituisce da quale materiale rotabile è composto un treno riportandone il tipo, la classe (se disponibile) e la quantità, i dati vengono presi dalla vista “composizione\_treno”.

**CREATE** **PROCEDURE** `composizione\_treno` **(IN** var\_treno **INT)**

**BEGIN**

**set** **transaction** **read** **only;**

**set** **transaction** **isolation** **level** **read** committed**;**

**start** **transaction;**

**select** treno\_matricola Matricola**,** Tipo**,** Classe**,** N\_Vagoni Quantita

**from** composizione\_treno

**where** Treno\_matricola **=** var\_treno**;**

**commit;**

**END**

**configurazione\_treno**

Procedura che restituisce la configurazione di un treno riportando nello specifico tutti i materiali da cui è composto con le loro specifiche.

**CREATE** **PROCEDURE** `configurazione\_treno` **(IN** var\_treno **INT)**

**BEGIN**

**set** **transaction** **read** **only;**

**set** **transaction** **isolation** **level** **read** committed**;**

**start** **transaction;**

**SELECT** Treno\_Matricola Matricola**,** Modello\_Marca\_Nome Marca**,** Modello\_Nome Modello**,** Tipo**,** Portata\_max**,** Num\_max\_passeggeri Pass\_max**,**Classe

**FROM** Materiale\_Rotabile**,** Modello

**where** Modello\_Nome**=**Nome **and** Modello\_Marca\_Nome**=**Marca\_Nome **and** Treno\_Matricola **=** var\_treno**;**

**commit;**

**END**

**convalida\_prenotazione**

Procedura che permette di convalidare una prenotazione aggiornando lo stato di quest’ultima da “Valida” ad “Utilizzata”.

**CREATE** **PROCEDURE** `convalida\_prenotazione` **(IN** var\_codice\_prenotazione **VARCHAR(**5**))**

**BEGIN**

**DECLARE** **EXIT** HANDLER **FOR** **sqlexception**

**BEGIN**

**rollback;**

resignal**;**

**END;**

**set** **transaction** **isolation** **level** serializable**;**

**UPDATE** `AziendaFerroviaria`**.**`Prenotazione`

**SET**

`Stato` **=** 'Utilizzata'

**WHERE** `Codice\_prenotazione` **=** var\_codice\_prenotazione**;**

**commit;**

**END**

**lavoratori\_disponibili**

Procedura che restituisce le informazioni di tutti i lavorato disponibili dati un giorno e gli orario di inizio e fine turno.

**CREATE** **PROCEDURE** `lavoratori\_disponibili` **(IN** var\_data **DATE,** **IN** var\_inizio

**TIME,** **IN** var\_fine **TIME)**

**BEGIN**

**set** **transaction** **read** **only;**

**set** **transaction** **isolation** **level** serializable**;**

**start** **transaction;**

**select** CF**,** Nome**,** Cognome

**from** Lavoratore

**where** CF **not** **in** **(**

**select** Lavoratore\_CF

**from** Copertura

**join** Turno **on** **(**Turno\_ID **=** ID**)**

**where** **Data** **=** var\_data **and** **((**var\_inizio **>** Ora\_inizio **and** var\_inizio **<**

Ora\_fine**)** **or** **(**var\_fine **>** Ora\_inizio **and** var\_fine **<** Ora\_fine**)**

**or** **(**var\_inizio **<=** Ora\_inizio **and** var\_fine **>=** Ora\_fine**))**

**union**

**select** Lavoratore\_CF

**from** Malattia

**join** Turno **on** **(**Turno\_ID **=** ID**)**

**where** **Data** **=** var\_data **and** **((**var\_inizio **>** Ora\_inizio **and** var\_inizio **<**

Ora\_fine**)** **or** **(**var\_fine **>** Ora\_inizio **and** var\_fine **<** Ora\_fine**)**

**or** **(**var\_inizio **<=** Ora\_inizio **and** var\_fine **>=** Ora\_fine**)**

**));**

**commit;**

**END**

**login**

Procedura che permette l’accesso alla base di dati con il proprio ruolo assegnato.

**CREATE** **PROCEDURE** `login` **(in** var\_username **varchar(**45**),** **in** var\_pass **varchar(**45**),** **out** var\_role **INT)**

**BEGIN**

**declare** var\_user\_role ENUM**(**'amministratore'**,** 'lavoratore'**,** 'controllore'**,**'manutentore'**);**

**select** `ruolo` **from** `login`

**where** `username` **=** var\_username

**and** `password` **=** var\_pass

**into** var\_user\_role**;**

-- See the corresponding enum in the client

**if** var\_user\_role **=** 'amministratore' **then**

**set** var\_role **=** 1**;**

elseif var\_user\_role **=** 'lavoratore' **then**

**set** var\_role **=** 2**;**

elseif var\_user\_role **=** 'controllore' **then**

**set** var\_role **=** 3**;**

elseif var\_user\_role **=** 'manutentore' **then**

**set** var\_role **=** 4**;**

**else**

**set** var\_role **=** 5**;**

**end** **if;**

**END**

**modifica\_tratta\_servizio**

Procedura che permette di modificare la tratta di un particolare servizio ferroviario.

**CREATE** **PROCEDURE** `modifica\_tratta\_servizio` **(IN** var\_servizio **INT,** **IN** var\_tratta **INT)**

**BEGIN**

**DECLARE** **EXIT** HANDLER **FOR** **sqlexception**

**BEGIN**

**rollback;**

resignal**;**

**END;**

**set** **transaction** **isolation** **level** serializable**;**

**start** **transaction;**

**UPDATE** `AziendaFerroviaria`**.**`Servizio\_Ferroviario`

**set** Tratta\_ID **=** var\_tratta

**where** ID **=** var\_servizio**;**

**commit;**

**END**

**modifica\_treno\_servizio**

Procedura che permette di modificare il treno che esegue uno specifico servizio ferroviario.

**CREATE** **PROCEDURE** `modifica\_treno\_servizio` **(IN** var\_servizio **INT,** **IN** var\_treno **VARCHAR(**4**))**

**BEGIN**

**DECLARE** **EXIT** HANDLER **FOR** **sqlexception**

**BEGIN**

**rollback;**

resignal**;**

**END;**

**set** **transaction** **isolation** **level** serializable**;**

**start** **transaction;**

**UPDATE** `AziendaFerroviaria`**.**`Servizio\_Ferroviario`

**set** Treno\_Matricola **=** var\_treno

**where** ID **=** var\_servizio**;**

**commit;**

**END**

**report\_lavoratore**

Procedura che permette di ottenere i turni con le relative informazioni per uno specifico lavoratore.

**CREATE** **PROCEDURE** `report\_lavoratore` **(IN** var\_lavoratore **VARCHAR(**16**))**

**BEGIN**

**set** **transaction** **read** **only;**

**set** **transaction** **isolation** **level** **read** committed**;**

**start** **transaction;**

**SELECT** Lavoratore\_CF CF**,** Turno**.Data,** Ora\_inizio**,** Ora\_fine**,** Servizio\_Ferroviario**.**ID Codice\_Servizio **FROM** Copertura**,**Turno**,**Servizio\_Ferroviario**,**Tratta

**where** Lavoratore\_CF**=**var\_lavoratore **and** Copertura**.**Turno\_ID**=**Turno**.**ID **and** Servizio\_Ferroviario**.**Turno\_ID**=**Turno**.**ID **and** Turno**.Data** **>=** **current\_date**

**and** Servizio\_Ferroviario**.**Tratta\_ID**=**Tratta**.**ID**;**

**commit;**

**END**

**report\_manutentore**

Procedura che permette di creare un report di manutenzione andando ad inserire i relativi dati all’interno della tabella report\_manutenzione.

**CREATE** **PROCEDURE** `report\_manutenzione` **(IN** var\_id **INT,** **IN** var\_data **DATE,** **IN** var\_info LONGTEXT**,** **IN** var\_materiale\_rotabile\_id **INT)**

**BEGIN**

**set** **transaction** **isolation** **level** **read** committed**;**

**start** **transaction;**

**INSERT** **INTO** `AziendaFerroviaria`**.**`Attivita\_Manutenzione`

**(**`**Data**`**,**

`Info`**,**

`Materiale\_Rotabile\_ID`**)**

**VALUES**

**(**var\_data**,**

var\_info**,**

var\_materiale\_rotabile\_id**);**

**set** var\_id **=** last\_insert\_id**();**

**commit;**

**END**

**richiesta\_malattia**

Procedura che permette di inserire uno specifico lavoratore in malattia rimuovendolo dal turno per cui si pone in malattia e inserendolo nella tabella “Malattia” in modo che non venga riassegnato per quel turno.

**CREATE** **PROCEDURE** `richiesta\_malattia` **(IN** var\_lavoratore **VARCHAR(**16**),** **IN** var\_turno **VARCHAR(**10**))**

**BEGIN**

**set** **transaction** **isolation** **level** repeatable **read;**

**start** **transaction;**

**insert** **into** Malattia

**select** Lavoratore\_CF**,** Turno\_ID

**from** Copertura

**where** Lavoratore\_CF **=** var\_lavoratore **and** Turno\_ID**=**

var\_turno**;**

**delete** **from** Copertura

**where** Lavoratore\_CF **=** var\_lavoratore **and** Turno\_ID **=** var\_turno**;**

**commit;**

**END**

**verifica\_prenotazione**

Procedura che verifica lo stato di una prenotazione.

**CREATE** **PROCEDURE** `verifica\_prenotazione` **(IN** var\_codice\_prenotazione **VARCHAR(**5**))**

**BEGIN**

**set** **transaction** **read** **only;**

**set** **transaction** **isolation** **level** **read** committed**;**

**start** **transaction;**

**SELECT** Stato **FROM** Prenotazione

**WHERE** Codice\_prenotazione **=** var\_codice\_prenotazione**;**

**commit;**

**END**

**fermate\_tratta**

Procedura che restituisce tutte le fermate con i relativi orari di partenza ed arrivo di una tratta.

**CREATE** **PROCEDURE** `fermate\_tratta` **(IN** var\_tratta **INT)**

**BEGIN**

**set** **transaction** **read** **only;**

**set** **transaction** **isolation** **level** serializable**;**

**start** **transaction;**

**SELECT** Stazione\_Nome\_Stazione Stazione**,**Orario\_Partenza Partenza**,** Orario\_Arrivo Arrivo **from** Fermata **where** Tratta\_ID **=** var\_tratta **order** **by** Sequenza**;**

**commit;**

**END**

**tratte\_giornaliere**

Procedura che restituisce tutte le tratte con i relativi capolinea di partenza ed arrivo.

**CREATE** **PROCEDURE** `tratte\_giornaliere` **()**

**BEGIN**

**set** **transaction** **read** **only;**

**set** **transaction** **isolation** **level** serializable**;**

**start** **transaction;**

**SELECT** Tratta\_ID Tratta**,** Capolinea\_partenza Da**,** Capolinea\_arrivo A**,** Orario\_Partenza Delle **FROM** Fermata**,**Tratta **where** Tratta\_ID**=**ID **and** Sequenza**=**1 **order** **by** Orario\_Partenza**;**

**commit;**

**END**

# Appendice: Implementazione

## Codice SQL per istanziare il database

-- MySQL Workbench Forward Engineering

**SET** @OLD\_UNIQUE\_CHECKS**=**@@UNIQUE\_CHECKS**,** UNIQUE\_CHECKS**=**0**;**

**SET** @OLD\_FOREIGN\_KEY\_CHECKS**=**@@FOREIGN\_KEY\_CHECKS**,** FOREIGN\_KEY\_CHECKS**=**0**;**

**SET** @OLD\_SQL\_MODE**=**@@SQL\_MODE**,** SQL\_MODE**=**'ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION'**;**

-- -----------------------------------------------------

-- Schema AziendaFerroviaria

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema AziendaFerroviaria

-- -----------------------------------------------------

**CREATE** **SCHEMA** **IF** **NOT** **EXISTS** `AziendaFerroviaria` **DEFAULT** **CHARACTER** **SET** utf8 **;**

**USE** `AziendaFerroviaria` **;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Treno`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Treno` **(**

`Matricola` **INT** **NOT** **NULL,**

`Data\_Acquisto` **DATE** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Matricola`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Turno`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Turno` **(**

`ID` **INT** **NOT** **NULL,**

`**Data**` **DATE** **NOT** **NULL,**

`Ora\_inizio` **TIME** **NOT** **NULL,**

`Ora\_fine` **TIME** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`ID`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Tratta`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Tratta` **(**

`ID` **INT** **NOT** **NULL,**

`Capolinea\_partenza` **VARCHAR(**45**)** **NOT** **NULL,**

`Capolinea\_arrivo` **VARCHAR(**45**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`ID`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Servizio\_Ferroviario`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Servizio\_Ferroviario` **(**

`ID` **INT** **NOT** **NULL,**

`Turno\_ID` **INT** **NOT** **NULL,**

`Treno\_Matricola` **INT** **NOT** **NULL,**

`Tratta\_ID` **INT** **NOT** **NULL,**

`**Data**` **DATE** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`ID`**),**

**INDEX** `fk\_Tratta\_Turno1\_idx` **(**`Turno\_ID` **ASC)** VISIBLE**,**

**INDEX** `fk\_Tratta\_Treno1\_idx` **(**`Treno\_Matricola` **ASC)** VISIBLE**,**

**INDEX** `fk\_Itinerario\_Tratta1\_idx` **(**`Tratta\_ID` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Tratta\_Turno1`

**FOREIGN** **KEY** **(**`Turno\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Turno` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Tratta\_Treno1`

**FOREIGN** **KEY** **(**`Treno\_Matricola`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Treno` **(**`Matricola`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Itinerario\_Tratta1`

**FOREIGN** **KEY** **(**`Tratta\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Tratta` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Provincia`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Provincia` **(**

`Sigla` **VARCHAR(**2**)** **NOT** **NULL,**

`Nome` **VARCHAR(**45**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Sigla`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Citta`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Citta` **(**

`Nome` **VARCHAR(**45**)** **NOT** **NULL,**

`Provincia\_Sigla` **VARCHAR(**2**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Nome`**,** `Provincia\_Sigla`**),**

**INDEX** `fk\_Citta\_Provincia1\_idx` **(**`Provincia\_Sigla` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Citta\_Provincia1`

**FOREIGN** **KEY** **(**`Provincia\_Sigla`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Provincia` **(**`Sigla`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Stazione`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Stazione` **(**

`Nome\_Stazione` **VARCHAR(**45**)** **NOT** **NULL,**

`Citta\_Nome` **VARCHAR(**45**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Nome\_Stazione`**),**

**INDEX** `fk\_Stazione\_Citta1\_idx` **(**`Citta\_Nome` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Stazione\_Citta1`

**FOREIGN** **KEY** **(**`Citta\_Nome`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Citta` **(**`Nome`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Marca`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Marca` **(**

`Nome` **VARCHAR(**45**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Nome`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Modello`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Modello` **(**

`Nome` **VARCHAR(**45**)** **NOT** **NULL,**

`Marca\_Nome` **VARCHAR(**45**)** **NOT** **NULL,**

`Tipo` **VARCHAR(**45**)** **NOT** **NULL,**

`Portata\_max` **INT** **NULL** **DEFAULT** 0**,**

`Num\_max\_passeggeri` **INT** **NULL** **DEFAULT** 0**,**

`Classe` **VARCHAR(**45**)** **NULL** **DEFAULT** '-'**,**

**PRIMARY** **KEY** **(**`Marca\_Nome`**,** `Nome`**),**

**INDEX** `fk\_Modello\_Marca1\_idx` **(**`Marca\_Nome` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Modello\_Marca1`

**FOREIGN** **KEY** **(**`Marca\_Nome`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Marca` **(**`Nome`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Materiale\_Rotabile`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Materiale\_Rotabile` **(**

`ID` **INT** **NOT** **NULL,**

`Treno\_Matricola` **INT** **NOT** **NULL,**

`Modello\_Marca\_Nome` **VARCHAR(**45**)** **NOT** **NULL,**

`Modello\_Nome` **VARCHAR(**45**)** **NOT** **NULL,**

**INDEX** `fk\_Vagone\_Treno1\_idx` **(**`Treno\_Matricola` **ASC)** VISIBLE**,**

**PRIMARY** **KEY** **(**`ID`**,** `Treno\_Matricola`**,** `Modello\_Marca\_Nome`**,** `Modello\_Nome`**),**

**INDEX** `fk\_Materiale\_Rotabile\_Modello1\_idx` **(**`Modello\_Marca\_Nome` **ASC,** `Modello\_Nome` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Vagone\_Treno1`

**FOREIGN** **KEY** **(**`Treno\_Matricola`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Treno` **(**`Matricola`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Materiale\_Rotabile\_Modello1`

**FOREIGN** **KEY** **(**`Modello\_Marca\_Nome` **,** `Modello\_Nome`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Modello` **(**`Marca\_Nome` **,** `Nome`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Attivita\_Manutenzione`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Attivita\_Manutenzione` **(**

`ID` **INT** **NOT** **NULL** AUTO\_INCREMENT**,**

`**Data**` **DATE** **NOT** **NULL,**

`Info` **VARCHAR(**500**)** **NOT** **NULL,**

`Materiale\_Rotabile\_ID` **INT** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`ID`**),**

**INDEX** `fk\_Attivita\_Manutenzione\_Materiale\_Rotabile1\_idx` **(**`Materiale\_Rotabile\_ID` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Attivita\_Manutenzione\_Materiale\_Rotabile1`

**FOREIGN** **KEY** **(**`Materiale\_Rotabile\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Materiale\_Rotabile` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Mansione`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Mansione` **(**

`ID` **INT** **NOT** **NULL,**

`Incarico` **VARCHAR(**45**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`ID`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`login`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`login` **(**

`username` **VARCHAR(**45**)** **NOT** **NULL,**

`password` **VARCHAR(**45**)** **NOT** **NULL,**

`ruolo` ENUM**(**'amministratore'**,** 'lavoratore'**,** 'controllore'**,** 'manutentore'**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`username`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Lavoratore`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Lavoratore` **(**

`CF` **VARCHAR(**16**)** **NOT** **NULL,**

`Mansione\_ID` **INT** **NOT** **NULL,**

`Nome` **VARCHAR(**45**)** **NOT** **NULL,**

`Cognome` **VARCHAR(**45**)** **NOT** **NULL,**

`Data\_Nascita` **DATE** **NOT** **NULL,**

`Luogo\_Nascita` **VARCHAR(**45**)** **NOT** **NULL,**

`login\_username` **VARCHAR(**45**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`CF`**),**

**INDEX** `fk\_Lavoratore\_Mansione1\_idx` **(**`Mansione\_ID` **ASC)** VISIBLE**,**

**INDEX** `fk\_Lavoratore\_login1\_idx` **(**`login\_username` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Lavoratore\_Mansione1`

**FOREIGN** **KEY** **(**`Mansione\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Mansione` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Lavoratore\_login1`

**FOREIGN** **KEY** **(**`login\_username`**)**

**REFERENCES** `AziendaFerroviaria`**.**`login` **(**`username`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Copertura`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Copertura` **(**

`Lavoratore\_CF` **VARCHAR(**16**)** **NOT** **NULL,**

`Turno\_ID` **INT** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Lavoratore\_CF`**,** `Turno\_ID`**),**

**INDEX** `fk\_Copertura\_Lavoratore1\_idx` **(**`Lavoratore\_CF` **ASC)** VISIBLE**,**

**INDEX** `fk\_Copertura\_Turno1\_idx` **(**`Turno\_ID` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Copertura\_Lavoratore1`

**FOREIGN** **KEY** **(**`Lavoratore\_CF`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Lavoratore` **(**`CF`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Copertura\_Turno1`

**FOREIGN** **KEY** **(**`Turno\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Turno` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Malattia`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Malattia` **(**

`Lavoratore\_CF` **VARCHAR(**16**)** **NOT** **NULL,**

`Turno\_ID` **INT** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Lavoratore\_CF`**,** `Turno\_ID`**),**

**INDEX** `fk\_Malattia\_Lavoratore1\_idx` **(**`Lavoratore\_CF` **ASC)** VISIBLE**,**

**INDEX** `fk\_Malattia\_Turno1\_idx` **(**`Turno\_ID` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Malattia\_Lavoratore1`

**FOREIGN** **KEY** **(**`Lavoratore\_CF`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Lavoratore` **(**`CF`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Malattia\_Turno1`

**FOREIGN** **KEY** **(**`Turno\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Turno` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Azienda`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Azienda` **(**

`PIVA` **VARCHAR(**11**)** **NOT** **NULL,**

`Nome` **VARCHAR(**45**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`PIVA`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Fattura`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Fattura` **(**

`Numero\_Fattura` **VARCHAR(**45**)** **NOT** **NULL,**

`Azienda\_PIVA\_Mittente` **VARCHAR(**11**)** **NOT** **NULL,**

`Azienda\_PIVA\_Destinatario` **VARCHAR(**11**)** **NOT** **NULL,**

`**Data**` **DATE** **NOT** **NULL,**

`Importo` **FLOAT** **NOT** **NULL,**

`IVA\_applicata` **INT** **NOT** **NULL,**

**INDEX** `fk\_Fattura\_Azienda1\_idx` **(**`Azienda\_PIVA\_Mittente` **ASC)** VISIBLE**,**

**INDEX** `fk\_Fattura\_Azienda2\_idx` **(**`Azienda\_PIVA\_Destinatario` **ASC)** VISIBLE**,**

**PRIMARY** **KEY** **(**`Numero\_Fattura`**,** `Azienda\_PIVA\_Mittente`**),**

**CONSTRAINT** `fk\_Fattura\_Azienda1`

**FOREIGN** **KEY** **(**`Azienda\_PIVA\_Mittente`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Azienda` **(**`PIVA`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Fattura\_Azienda2`

**FOREIGN** **KEY** **(**`Azienda\_PIVA\_Destinatario`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Azienda` **(**`PIVA`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Merce\_trasportata`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Merce\_trasportata` **(**

`ID` **INT** **NOT** **NULL,**

`Materiale\_Rotabile\_ID` **INT** **NOT** **NULL,**

`Massa` **INT** **NOT** **NULL,**

`Descrizione` **VARCHAR(**45**)** **NOT** **NULL,**

`Servizio\_Ferroviario\_ID` **INT** **NOT** **NULL,**

`Fattura\_Numero\_Fattura` **VARCHAR(**45**)** **NOT** **NULL,**

`Fattura\_Azienda\_PIVA\_Mittente` **VARCHAR(**11**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`ID`**),**

**INDEX** `fk\_Merce\_trasportata\_Materiale\_Rotabile1\_idx` **(**`Materiale\_Rotabile\_ID` **ASC)** VISIBLE**,**

**INDEX** `fk\_Merce\_trasportata\_Servizio\_Ferroviario1\_idx` **(**`Servizio\_Ferroviario\_ID` **ASC)** VISIBLE**,**

**INDEX** `fk\_Merce\_trasportata\_Fattura1\_idx` **(**`Fattura\_Numero\_Fattura` **ASC,** `Fattura\_Azienda\_PIVA\_Mittente` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Merce\_trasportata\_Materiale\_Rotabile1`

**FOREIGN** **KEY** **(**`Materiale\_Rotabile\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Materiale\_Rotabile` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Merce\_trasportata\_Servizio\_Ferroviario1`

**FOREIGN** **KEY** **(**`Servizio\_Ferroviario\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Servizio\_Ferroviario` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Merce\_trasportata\_Fattura1`

**FOREIGN** **KEY** **(**`Fattura\_Numero\_Fattura` **,** `Fattura\_Azienda\_PIVA\_Mittente`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Fattura` **(**`Numero\_Fattura` **,** `Azienda\_PIVA\_Mittente`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Posto`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Posto` **(**

`Num\_posto` **VARCHAR(**3**)** **NOT** **NULL,**

`Materiale\_Rotabile\_ID` **INT** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Num\_posto`**,** `Materiale\_Rotabile\_ID`**),**

**INDEX** `fk\_Posto\_Materiale\_Rotabile1\_idx` **(**`Materiale\_Rotabile\_ID` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Posto\_Materiale\_Rotabile1`

**FOREIGN** **KEY** **(**`Materiale\_Rotabile\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Materiale\_Rotabile` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Passeggero`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Passeggero` **(**

`CF` **VARCHAR(**16**)** **NOT** **NULL,**

`Nome` **VARCHAR(**45**)** **NOT** **NULL,**

`Cognome` **VARCHAR(**45**)** **NOT** **NULL,**

`Data\_nascita` **DATE** **NOT** **NULL,**

`Carta\_di\_credito` **VARCHAR(**16**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`CF`**))**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Prenotazione`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Prenotazione` **(**

`Codice\_prenotazione` **VARCHAR(**5**)** **NOT** **NULL,**

`Posto\_Num\_posto` **VARCHAR(**3**)** **NOT** **NULL,**

`Passeggero\_CF` **VARCHAR(**16**)** **NOT** **NULL,**

`Stato` **VARCHAR(**45**)** **NOT** **NULL,**

`Servizio\_Ferroviario\_ID` **INT** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Codice\_prenotazione`**),**

**INDEX** `fk\_Prenotazione\_Posto1\_idx` **(**`Posto\_Num\_posto` **ASC)** VISIBLE**,**

**INDEX** `fk\_Prenotazione\_Passeggero1\_idx` **(**`Passeggero\_CF` **ASC)** VISIBLE**,**

**INDEX** `fk\_Prenotazione\_Servizio\_Ferroviario1\_idx` **(**`Servizio\_Ferroviario\_ID` **ASC)** VISIBLE**,**

**CONSTRAINT** `fk\_Prenotazione\_Posto1`

**FOREIGN** **KEY** **(**`Posto\_Num\_posto`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Posto` **(**`Num\_posto`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Prenotazione\_Passeggero1`

**FOREIGN** **KEY** **(**`Passeggero\_CF`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Passeggero` **(**`CF`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Prenotazione\_Servizio\_Ferroviario1`

**FOREIGN** **KEY** **(**`Servizio\_Ferroviario\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Servizio\_Ferroviario` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

-- -----------------------------------------------------

-- Table `AziendaFerroviaria`.`Fermata`

-- -----------------------------------------------------

**CREATE** **TABLE** **IF** **NOT** **EXISTS** `AziendaFerroviaria`**.**`Fermata` **(**

`Tratta\_ID` **INT** **NOT** **NULL,**

`Stazione\_Nome\_Stazione` **VARCHAR(**45**)** **NOT** **NULL,**

`Orario\_Partenza` **TIME** **NULL,**

`Orario\_Arrivo` **TIME** **NULL,**

`Sequenza` **INT** **NOT** **NULL,**

**PRIMARY** **KEY** **(**`Tratta\_ID`**,** `Stazione\_Nome\_Stazione`**),**

**INDEX** `fk\_Fermata\_Stazione1\_idx` **(**`Stazione\_Nome\_Stazione` **ASC)** INVISIBLE**,**

**CONSTRAINT** `fk\_Fermata\_Tratta1`

**FOREIGN** **KEY** **(**`Tratta\_ID`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Tratta` **(**`ID`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**,**

**CONSTRAINT** `fk\_Fermata\_Stazione1`

**FOREIGN** **KEY** **(**`Stazione\_Nome\_Stazione`**)**

**REFERENCES** `AziendaFerroviaria`**.**`Stazione` **(**`Nome\_Stazione`**)**

**ON** **DELETE** **NO** ACTION

**ON** **UPDATE** **NO** ACTION**)**

ENGINE **=** InnoDB**;**

**CREATE** **USER** 'amministratore' **IDENTIFIED** **BY** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`aggiungi\_lavoratore` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`aggiungi\_azienda` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`aggiungi\_passeggero` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`aggiungi\_prenotazione` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`aggiungi\_spedizione` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`aggiungi\_turno` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`lavoratori\_disponibili` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`richiesta\_malattia` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`assegna\_turno` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`configurazione\_treno` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`composizione\_treno` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`modifica\_treno\_servizio` **TO** 'amministratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`modifica\_tratta\_servizio` **TO** 'amministratore'**;**

**CREATE** **USER** 'lavoratore' **IDENTIFIED** **BY** 'lavoratore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`report\_lavoratore` **TO** 'lavoratore'**;**

**CREATE** **USER** 'controllore' **IDENTIFIED** **BY** 'controllore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`verifica\_prenotazione` **TO** 'controllore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`convalida\_prenotazione` **TO** 'controllore'**;**

**CREATE** **USER** 'manutentore' **IDENTIFIED** **BY** 'manutentore'**;**

**GRANT** **EXECUTE** **ON** **procedure** `AziendaFerroviaria`**.**`report\_manutenzione` **TO** 'manutentore'**;**

**SET** SQL\_MODE**=**@OLD\_SQL\_MODE**;**

**SET** FOREIGN\_KEY\_CHECKS**=**@OLD\_FOREIGN\_KEY\_CHECKS**;**

**SET** UNIQUE\_CHECKS**=**@OLD\_UNIQUE\_CHECKS**;**

## Codice del Front-End

**main.c**

Permette ad un utente di collegarsi al database e di effettuare l’accesso per ottenere i privilegi assegnatoli dal proprio ruolo.

#include <stdio.h>

#include <mysql.h>

#include <stdlib.h>

#include <string.h>

#include "defines.h"

#define ANSI\_COLOR\_RED "\x1b[31m"

#define ANSI\_COLOR\_GREEN "\x1b[32m"

#define ANSI\_COLOR\_YELLOW "\x1b[33m"

#define ANSI\_COLOR\_BLUE "\x1b[34m"

#define ANSI\_COLOR\_MAGENTA "\x1b[35m"

#define ANSI\_COLOR\_CYAN "\x1b[36m"

#define ANSI\_COLOR\_RESET "\x1b[0m"

static MYSQL **\***conn**;**

**typedef** enum **{**

ADMIN **=** 1**,**

LAVORATORE**,**

CONTROLLORE**,**

MANUTENTORE**,**

FAILED\_LOGIN

**}** role\_t**;**

static role\_t attempt\_login**(**MYSQL **\***conn**,** char **\***username**,** char **\***password**)** **{**

MYSQL\_STMT **\***login\_procedure**;**

MYSQL\_BIND param**[**3**];** // Used both for input and output

int role **=** 0**;**

**if(!**setup\_prepared\_stmt**(&**login\_procedure**,** "call login(?, ?, ?)"**,** conn**))** **{**

print\_stmt\_error**(**login\_procedure**,** "Unable to initialize login statement\n"**);**

**goto** err2**;**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;** // IN

param**[**0**].**buffer **=** username**;**

param**[**0**].**buffer\_length **=** strlen**(**username**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;** // IN

param**[**1**].**buffer **=** password**;**

param**[**1**].**buffer\_length **=** strlen**(**password**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;** // OUT

param**[**2**].**buffer **=** **&**role**;**

param**[**2**].**buffer\_length **=** **sizeof(**role**);**

**if** **(**mysql\_stmt\_bind\_param**(**login\_procedure**,** param**)** **!=** 0**)** **{** // Note \_param

print\_stmt\_error**(**login\_procedure**,** "Could not bind parameters for login"**);**

**goto** err**;**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**login\_procedure**)** **!=** 0**)** **{**

print\_stmt\_error**(**login\_procedure**,** "Could not execute login procedure"**);**

**goto** err**;**

**}**

// Prepare output parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;** // OUT

param**[**0**].**buffer **=** **&**role**;**

param**[**0**].**buffer\_length **=** **sizeof(**role**);**

**if(**mysql\_stmt\_bind\_result**(**login\_procedure**,** param**))** **{**

print\_stmt\_error**(**login\_procedure**,** "Could not retrieve output parameter"**);**

**goto** err**;**

**}**

// Retrieve output parameter

**if(**mysql\_stmt\_fetch**(**login\_procedure**))** **{**

print\_stmt\_error**(**login\_procedure**,** "Could not buffer results"**);**

**goto** err**;**

**}**

mysql\_stmt\_close**(**login\_procedure**);**

**return** role**;**

err**:**

mysql\_stmt\_close**(**login\_procedure**);**

err2**:**

**return** FAILED\_LOGIN**;**

**}**

int main**(**void**){**

char username**[**128**];**

char password**[**128**];**

conn **=** mysql\_init **(NULL);**

**if** **(**conn **==** **NULL)** **{**

fprintf **(**stderr**,** "mysql\_init() failed (probably out of memory)\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

**if** **(**mysql\_real\_connect**(**conn**,** "localhost"**,** "login"**,** "Lineagialla1!"**,**"AziendaFerroviaria"**,** 3306**,** **NULL,** CLIENT\_MULTI\_STATEMENTS **|** CLIENT\_MULTI\_RESULTS**)** **==** **NULL)** **{**

fprintf **(**stderr**,** "mysql\_real\_connect() failed\n"**);**

mysql\_close **(**conn**);**

exit**(**1**);**

**}**

printf**(**ANSI\_COLOR\_CYAN "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

printf**(** "\*\*\* Azienda Ferroviaria DB \*\*\*\n"**);**

printf**(** "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ANSI\_COLOR\_RESET "\n"**);**

printf**(**ANSI\_COLOR\_GREEN "Benvenuto, inserisci i tuoi dati per accedere:\n\n"ANSI\_COLOR\_RESET**);**

printf**(**"Username: "**);**

getInput**(**128**,** username**,** false**);**

printf**(**"Password: "**);**

getInput**(**128**,** password**,** true**);**

int role **=** attempt\_login**(**conn**,** username**,** password**);**

**switch(**role**)** **{**

**case** ADMIN**:**

admin**(**conn**,** username**);**

**break;**

**case** LAVORATORE**:**

lavoratore**(**conn**,** username**);**

**break;**

**case** CONTROLLORE**:**

controllore**(**conn**,** username**);**

**break;**

**case** MANUTENTORE**:**

manutentore**(**conn**,** username**);**

**break;**

**default:**

printf**(**ANSI\_COLOR\_RED"Dati non corretti!\n"ANSI\_COLOR\_RESET**);**

abort**();**

**}**

mysql\_close**(**conn**);**

**return** 0**;**

**}**

**admin.c**

Utilizzato per l’accesso come amministratore e per far accedere all’utente a tutte le procedure del suo ruolo.

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <signal.h>

#include <errno.h>

#include "defines.h"

#define ANSI\_COLOR\_RED "\x1b[31m"

#define ANSI\_COLOR\_GREEN "\x1b[32m"

#define ANSI\_COLOR\_YELLOW "\x1b[33m"

#define ANSI\_COLOR\_BLUE "\x1b[34m"

#define ANSI\_COLOR\_MAGENTA "\x1b[35m"

#define ANSI\_COLOR\_CYAN "\x1b[36m"

#define ANSI\_COLOR\_RESET "\x1b[0m"

char command**[**20**];**

static void aggiungi\_lavoratore**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**8**];**

char cf**[**16**];**

int mansione\_id**;**

char name**[**45**];**

char surname**[**45**];**

MYSQL\_TIME **\***date**;** //Date of birth

char pob**[**45**];** //Place of birth

date **=** malloc**(sizeof(**MYSQL\_TIME**));**

char username**[**45**];**

char password**[**45**];**

char passwordtmp**[**45**];**

printf**(**"Aggiungi lavoratore\n"**);**

// Get the required information

printf**(**"CF: "**);**

**if(**getInput**(**16**,** cf**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"ID Mansione: "**);**

**if(**getInteger**(&**mansione\_id**)** **<** 0**){**

**return;**

**}**

printf**(**"Nome: "**);**

**if(**getInput**(**45**,** name**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Cognome: "**);**

**if(**getInput**(**45**,** surname**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Luogo di nascita: "**);**

**if(**getInput**(**45**,** pob**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Inserisci data di nascita(formato dd/mm/yyyy): "**);**

**if(**getDate**(**date**)** **<** 0**){**

**return;**

**}**

printf**(**"Username: "**);**

**if(**getInput**(**45**,** username**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Password: "**);**

**if(**getInput**(**45**,** password**,** true**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Ripeti password: "**);**

**if(**getInput**(**45**,** passwordtmp**,** true**)** **<** 0**)** **{**

**return;**

**}**

**if(**strcmp**(**password**,** passwordtmp**))** **{**

printf**(**"Le password non corrispondono\n"**);**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call aggiungi\_lavoratore(?, ?, ?, ?, ?, ?, ?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** cf**;**

param**[**0**].**buffer\_length **=** strlen**(**cf**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**1**].**buffer **=** **&**mansione\_id**;**

param**[**1**].**buffer\_length **=** **sizeof(**mansione\_id**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**2**].**buffer **=** name**;**

param**[**2**].**buffer\_length **=** strlen**(**name**);**

param**[**3**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**3**].**buffer **=** surname**;**

param**[**3**].**buffer\_length **=** strlen**(**surname**);**

param**[**4**].**buffer\_type **=** MYSQL\_TYPE\_DATE**;**

param**[**4**].**buffer **=** date**;**

param**[**4**].**buffer\_length **=** **sizeof(\***date**);**

param**[**5**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**5**].**buffer **=** pob**;**

param**[**5**].**buffer\_length **=** strlen**(**pob**);**

param**[**6**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**6**].**buffer **=** username**;**

param**[**6**].**buffer\_length **=** strlen**(**username**);**

param**[**7**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**7**].**buffer **=** password**;**

param**[**7**].**buffer\_length **=** strlen**(**password**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'aggiunta del lavoratore."**);**

**}** **else** **{**

printf**(**"Lavoratore %s aggiunto correttamente\n"**,** username**);**

**}**

free**(**date**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void aggiungi\_turno**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**4**];**

int id**;**

MYSQL\_TIME **\***date**;**

date **=** malloc**(sizeof(**MYSQL\_TIME**));**

MYSQL\_TIME **\***start**;**

MYSQL\_TIME **\***end**;**

start **=** malloc**(sizeof(**MYSQL\_TIME**));**

end **=** malloc**(sizeof(**MYSQL\_TIME**));**

printf**(**"Aggiungi turno\n"**);**

// Get the required information

printf**(**"\nID Turno: "**);**

**if(**getInteger**(&**id**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Inserisci data(formato dd/mm/yyyy): "**);**

**if(**getDate**(**date**)** **<** 0**){**

**return;**

**}**

printf**(**"Inserisci orario di inizio(formato hh:mm): "**);**

**if(**getTime**(**start**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Inserisci orario di fine(formato hh:mm): "**);**

**if(**getTime**(**end**)** **<** 0**)** **{**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call aggiungi\_turno(?, ?, ?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**0**].**buffer **=** **&**id**;**

param**[**0**].**buffer\_length **=** **sizeof(**id**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_DATE**;**

param**[**1**].**buffer **=** date**;**

param**[**1**].**buffer\_length **=** **sizeof(\***date**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_TIME**;**

param**[**2**].**buffer **=** start**;**

param**[**2**].**buffer\_length **=** **sizeof(\***start**);**

param**[**3**].**buffer\_type **=** MYSQL\_TYPE\_TIME**;**

param**[**3**].**buffer **=** end**;**

param**[**3**].**buffer\_length **=** **sizeof(\***end**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'aggiunta del turno."**);**

**}** **else** **{**

printf**(**"Turno %d aggiunto correttamente\n"**,** id**);**

**}**

free**(**start**);**

free**(**end**);**

free**(**date**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void assegna\_turno**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**2**];**

int turno**;**

char cf**[**17**];**

printf**(**"Assegna turno\n"**);**

// Get the required information

printf**(**"CF Lavoratore: "**);**

**if(**getInput**(**17**,** cf**,** false**)** **<** 0**){**

**return;**

**}**

printf**(**"\nID Turno: "**);**

**if(**getInteger**(&**turno**)** **<** 0**){**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call assegna\_turno(?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** cf**;**

param**[**0**].**buffer\_length **=** strlen**(**cf**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**1**].**buffer **=** **&**turno**;**

param**[**1**].**buffer\_length **=** **sizeof(**turno**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nella assegnazione del turno."**);**

**}** **else** **{**

printf**(**"Lavoratore %s assegnato al turno %d.\n"**,** cf**,** turno**);**

**}**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void aggiungi\_azienda**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**2**];**

char nome**[**45**];**

char piva**[**12**];**

printf**(**"Aggiungi Azienda\n"**);**

// Get the required information

printf**(**"\nPartita IVA: "**);**

**if(**getInput**(**12**,** piva**,** false**)** **<** 0**){**

**return;**

**}**

printf**(**"Nome azienda: "**);**

**if(**getInput**(**45**,** nome**,** false**)** **<** 0**){**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call aggiungi\_azienda(?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** piva**;**

param**[**0**].**buffer\_length **=** strlen**(**piva**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**1**].**buffer **=** nome**;**

param**[**1**].**buffer\_length **=** strlen**(**nome**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'aggiunta dell'azienda."**);**

**}** **else** **{**

printf**(**"Azienda %s aggiunta correttamente.\n"**,** nome**);**

**}**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void aggiungi\_passeggero**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**5**];**

char cf**[**16**];**

char carta**[**16**];**

char name**[**45**];**

char surname**[**45**];**

MYSQL\_TIME **\***date**;**//Date of birth

date **=** malloc**(sizeof(**MYSQL\_TIME**));**

printf**(**"Aggiungi Passeggero\n"**);**

// Get the required information

printf**(**"CF: "**);**

**if(**getInput**(**16**,** cf**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Nome: "**);**

**if(**getInput**(**45**,** name**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Cognome: "**);**

**if(**getInput**(**45**,** surname**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Data di nascita(formato dd/mm/yyyy): "**);**

**if(**getDate**(**date**)** **<** 0**){**

**return;**

**}**

printf**(**"Carta di credito: "**);**

**if(**getInput**(**16**,** carta**,** false**)** **<** 0**)** **{**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call aggiungi\_passeggero(?, ?, ?, ?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** cf**;**

param**[**0**].**buffer\_length **=** strlen**(**cf**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**1**].**buffer **=** name**;**

param**[**1**].**buffer\_length **=** strlen**(**name**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**2**].**buffer **=** surname**;**

param**[**2**].**buffer\_length **=** strlen**(**surname**);**

param**[**3**].**buffer\_type **=** MYSQL\_TYPE\_DATE**;**

param**[**3**].**buffer **=** date**;**

param**[**3**].**buffer\_length **=** **sizeof(\***date**);**

param**[**4**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**4**].**buffer **=** carta**;**

param**[**4**].**buffer\_length **=** strlen**(**carta**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'aggiunta del passeggero."**);**

**}** **else** **{**

printf**(**"Passeggero %s aggiunto correttamente\n"**,** cf**);**

**}**

free**(**date**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void aggiungi\_prenotazione**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**4**];**

char codice**[**6**];**

char cf**[**17**];**

char posto**[**4**];**

int servizio**;**

printf**(**"Aggiungi Prenotazione\n"**);**

// Get the required information

printf**(**"Codice prenotazione (5 cifre): "**);**

**if(**getInput**(**6**,** codice**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Numero posto (3 cifre): "**);**

**if(**getInput**(**4**,** posto**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Codice fiscale passeggero: "**);**

**if(**getInput**(**17**,** cf**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"ID Servizio Ferroviario: "**);**

**if(**getInteger**(&**servizio**)** **<** 0**){**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call aggiungi\_prenotazione(?, ?, ?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** codice**;**

param**[**0**].**buffer\_length **=** strlen**(**codice**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**1**].**buffer **=** posto**;**

param**[**1**].**buffer\_length **=** strlen**(**posto**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**2**].**buffer **=** cf**;**

param**[**2**].**buffer\_length **=** strlen**(**cf**);**

param**[**3**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**3**].**buffer **=** **&**servizio**;**

param**[**3**].**buffer\_length **=** **sizeof(**servizio**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'aggiunta della prenotazione."**);**

**}** **else** **{**

printf**(**"Prenotazione %s per il passeggero %s aggiunta correttamente\n"**,** codice**,** cf**);**

**}**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void aggiungi\_servizio**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**5**];**

char treno**[**4**];**

int servizio**;**

int turno**;**

int tratta**;**

MYSQL\_TIME **\***date**;**

date **=** malloc**(sizeof(**MYSQL\_TIME**));**

printf**(**"Aggiungi Servizio\n"**);**

// Get the required informationù

printf**(**"ID Servizio Ferroviario: "**);**

**if(**getInteger**(&**servizio**)** **<** 0**){**

**return;**

**}**

printf**(**"Matricola treno (4 cifre): "**);**

**if(**getInput**(**5**,** treno**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"ID Tratta: "**);**

**if(**getInteger**(&**tratta**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"ID Turno: "**);**

**if(**getInteger**(&**turno**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Data (formato dd/mm/yyyy): "**);**

**if(**getDate**(**date**)** **<** 0**){**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call aggiungi\_servizio(?, ?, ?, ?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**0**].**buffer **=** **&**servizio**;**

param**[**0**].**buffer\_length **=** **sizeof(**servizio**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**1**].**buffer **=** **&**turno**;**

param**[**1**].**buffer\_length **=** **sizeof(**turno**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**2**].**buffer **=** treno**;**

param**[**2**].**buffer\_length **=** strlen**(**treno**);**

param**[**3**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**3**].**buffer **=** **&**tratta**;**

param**[**3**].**buffer\_length **=** **sizeof(**tratta**);**

param**[**4**].**buffer\_type **=** MYSQL\_TYPE\_DATE**;**

param**[**4**].**buffer **=** date**;**

param**[**4**].**buffer\_length **=** **sizeof(\***date**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'aggiunta del servizio ferroviario."**);**

**}** **else** **{**

printf**(**"Servizio ferroviario %d sulla tratta %d aggiunto correttamente\n"**,** servizio**,** tratta**);**

**}**

free**(**date**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void aggiungi\_spedizione**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**7**];**

char mittente**[**11**];**

char fattura**[**45**];**

char descrizione**[**120**];**

int massa**;**

int servizio**;**

int vagone**;**

int id**;**

printf**(**"Aggiungi Spedizione\n"**);**

// Get the required information

printf**(**"Codice spedizione: "**);**

**if(**getInteger**(&**id**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Partita IVA mittente: "**);**

**if(**getInput**(**11**,** mittente**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Numero fattura: "**);**

**if(**getInput**(**45**,** fattura**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"ID Servizio Ferroviario: "**);**

**if(**getInteger**(&**servizio**)** **<** 0**){**

**return;**

**}**

printf**(**"ID Vagone: "**);**

**if(**getInteger**(&**vagone**)** **<** 0**){**

**return;**

**}**

printf**(**"Descrizione contenuto: "**);**

**if(**getInput**(**120**,** descrizione**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"Massa: "**);**

**if(**getInteger**(&**massa**)** **<** 0**){**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call aggiungi\_spedizione(?, ?, ?, ?, ?, ?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**0**].**buffer **=** **&**id**;**

param**[**0**].**buffer\_length **=** **sizeof(**id**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**1**].**buffer **=** mittente**;**

param**[**1**].**buffer\_length **=** strlen**(**mittente**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**2**].**buffer **=** **&**servizio**;**

param**[**2**].**buffer\_length **=** **sizeof(**servizio**);**

param**[**3**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**3**].**buffer **=** **&**vagone**;**

param**[**3**].**buffer\_length **=** **sizeof(**vagone**);**

param**[**4**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**4**].**buffer **=** **&**massa**;**

param**[**4**].**buffer\_length **=** **sizeof(**massa**);**

param**[**5**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**5**].**buffer **=** descrizione**;**

param**[**5**].**buffer\_length **=** strlen**(**descrizione**);**

param**[**6**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**6**].**buffer **=** fattura**;**

param**[**6**].**buffer\_length **=** strlen**(**fattura**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'aggiunta della spedizione."**);**

**}** **else** **{**

printf**(**"Spedizione %d aggiunta correttamente\n"**,** id**);**

**}**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void richiesta\_malattia**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**2**];**

char cf**[**16**];**

int turno**;**

printf**(**"\nRichiesta malattia\n"**);**

// Get the required information

printf**(**"\nCF Lavoratore: "**);**

**if(**getInput**(**16**,** cf**,** false**)** **<** 0**)** **{**

**return;**

**}**

printf**(**"ID Turno: "**);**

**if(**getInteger**(&**turno**)** **<** 0**)** **{**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call richiesta\_malattia(?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** cf**;**

param**[**0**].**buffer\_length **=** strlen**(**cf**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**1**].**buffer **=** **&**turno**;**

param**[**1**].**buffer\_length **=** **sizeof(**turno**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nella richiesta di malattia."**);**

**}** **else** **{**

printf**(**"Lavoratore %s correttamente inserito in malattia.\n"**,** cf**);**

**}**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void lavoratori\_disponibili**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**3**];**

MYSQL\_TIME **\***date**;**

MYSQL\_TIME **\***start**;**

MYSQL\_TIME **\***end**;**

int status**;**

date **=** malloc**(sizeof(**MYSQL\_TIME**));**

start **=** malloc**(sizeof(**MYSQL\_TIME**));**

end **=** malloc**(sizeof(**MYSQL\_TIME**));**

printf**(**"\n"**);**

printf**(**"Ricerca lavoratori disponibili\n"**);**

// Get the required information

printf**(**"Inserisci data turno(formato dd/mm/yyyy): "**);**

**if(**getDate**(**date**)** **<** 0**)** **return;**

printf**(**"Inserisci orario di inizio(formato hh:mm): "**);**

**if(**getTime**(**start**)** **<** 0**)** **return;**

printf**(**"Inserisci orario di fine(formato hh:mm): "**);**

**if(**getTime**(**end**)** **<** 0**)** **return;**

getchar**();**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call lavoratori\_disponibili(?, ?, ?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_DATE**;**

param**[**0**].**buffer **=** date**;**

param**[**0**].**buffer\_length **=** **sizeof(\***date**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_TIME**;**

param**[**1**].**buffer **=** start**;**

param**[**1**].**buffer\_length **=** **sizeof(\***start**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_TIME**;**

param**[**2**].**buffer **=** end**;**

param**[**2**].**buffer\_length **=** **sizeof(\***end**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nella ricerca di lavoratori disponibili."**);**

**}**

char buff**[**100**];**

sprintf**(**buff**,** "Lavoratori disponibili il giorno %d/%d/%d %u:%u-%u:%u"**,** date**->**day**,** date**->**month**,** date**->**year**,** start**->**hour**,** start**->**minute**,** end**->**hour**,** end**->**minute**);**

**do** **{**

dump\_result\_set**(**conn**,** prepared\_stmt**,** buff**);**

status **=** mysql\_stmt\_next\_result**(**prepared\_stmt**);**

**if** **(**status **>** 0**)**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Unexpected condition"**,** true**);**

**}** **while** **(**status **==** 0**);**

free**(**start**);**

free**(**end**);**

free**(**date**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void modifica\_servizio**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**2**];**

int servizio**;**

int tratta**;**

printf**(**"\nModifica associazione Servizio a Tratta\n"**);**

// Get the required information

printf**(**"ID Servizio Ferroviario: "**);**

**if(**getInteger**(&**servizio**)** **<** 0**){**

**return;**

**}**

printf**(**"ID Tratta: "**);**

**if(**getInteger**(&**tratta**)** **<** 0**){**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call modifica\_tratta\_servizio(?,?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**0**].**buffer **=** **&**servizio**;**

param**[**0**].**buffer\_length **=** **sizeof(**servizio**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**1**].**buffer **=** **&**tratta**;**

param**[**1**].**buffer\_length **=** **sizeof(**tratta**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nella modifica associazione Servizio a Tratta."**);**

**}else** **{**

printf**(**"Modifica effettuata con successo.\n"**);**

**}**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void modifica\_treno**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**2**];**

int servizio**;**

char treno**[**4**];**

printf**(**"\nModifica associazione Treno a Servizio\n"**);**

// Get the required information

printf**(**"ID Servizio Ferroviario: "**);**

**if(**getInteger**(&**servizio**)** **<** 0**){**

**return;**

**}**

printf**(**"Matricola treno (4 cifre): "**);**

**if(**getInput**(**5**,** treno**,** false**)** **<** 0**)** **{**

**return;**

**}**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call modifica\_treno\_servizio(?,?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**0**].**buffer **=** **&**servizio**;**

param**[**0**].**buffer\_length **=** **sizeof(**servizio**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**1**].**buffer **=** treno**;**

param**[**1**].**buffer\_length **=** strlen**(**treno**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nella modifica associazione treno a servizio."**);**

**}else** **{**

printf**(**"Modifica effettuata con successo.\n"**);**

**}**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void configurazione**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**1**];**

char treno**[**5**];**

int status**;**

printf**(**"\nConfigurazione Treno\n"**);**

// Get the required information

printf**(**"Matricola Treno: "**);**

**if(**getInput**(**5**,** treno**,** false**)** **<** 0**)** **return;**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call configurazione\_treno(?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** treno**;**

param**[**0**].**buffer\_length **=** strlen**(**treno**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore, impossibile ottenere la composizione del treno."**);**

**}**

**do** **{**

dump\_result\_set**(**conn**,** prepared\_stmt**,** ""**);**

status **=** mysql\_stmt\_next\_result**(**prepared\_stmt**);**

**if** **(**status **>** 0**)**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Unexpected condition"**,** true**);**

**}** **while** **(**status **==** 0**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void composizione**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**1**];**

char treno**[**5**];**

int status**;**

printf**(**"\nComposizione Treno\n"**);**

// Get the required information

printf**(**"Matricola Treno: "**);**

**if(**getInput**(**5**,** treno**,** false**)** **<** 0**)** **return;**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call composizione\_treno(?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** treno**;**

param**[**0**].**buffer\_length **=** strlen**(**treno**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore, impossibile ottenere la configurazione del treno."**);**

**}**

**do** **{**

dump\_result\_set**(**conn**,** prepared\_stmt**,** ""**);**

status **=** mysql\_stmt\_next\_result**(**prepared\_stmt**);**

**if** **(**status **>** 0**)**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Unexpected condition"**,** true**);**

**}** **while** **(**status **==** 0**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void fermate**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**1**];**

int tratta**;**

int status**;**

printf**(**"\nFermate della tratta\n"**);**

// Get the required information

printf**(**"ID Tratta: "**);**

**if(**getInteger**(&**tratta**)** **<** 0**)**

**return;**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call fermate\_tratta(?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**0**].**buffer **=** **&**tratta**;**

param**[**0**].**buffer\_length **=** **sizeof(**tratta**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore, impossibile ottenere le fermate della tratta."**);**

**}**

**do** **{**

dump\_result\_set**(**conn**,** prepared\_stmt**,** ""**);**

status **=** mysql\_stmt\_next\_result**(**prepared\_stmt**);**

**if** **(**status **>** 0**)**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Unexpected condition"**,** true**);**

**}** **while** **(**status **==** 0**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void tratte**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

int status**;**

printf**(**"\nTratte giornaliere\n"**);**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call tratte\_giornaliere()"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore, impossibile ottenere le fermate della tratta."**);**

**}**

**do** **{**

dump\_result\_set**(**conn**,** prepared\_stmt**,** ""**);**

status **=** mysql\_stmt\_next\_result**(**prepared\_stmt**);**

**if** **(**status **>** 0**)**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Unexpected condition"**,** true**);**

**}** **while** **(**status **==** 0**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

void admin**(**MYSQL **\***conn**,** char **\***username**)** **{**

printf**(**"\033[2J\033[H"**);**

**if(**mysql\_change\_user**(**conn**,** "admin"**,** "Lineagialla1!"**,** "AziendaFerroviaria"**))** **{**

fprintf**(**stderr**,** "mysql\_change\_user() failed\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

struct sigaction act**,** sa**;**

memset **(&**act**,** '\0'**,** **sizeof(**act**));**

printf**(**ANSI\_COLOR\_YELLOW "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" ANSI\_COLOR\_RESET**);**

printf**(**ANSI\_COLOR\_YELLOW "\*\*\*Connected as admin\*\*\*\n" ANSI\_COLOR\_RESET**);**

printf**(**ANSI\_COLOR\_YELLOW "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" ANSI\_COLOR\_RESET**);**

**while(**1**)** **{**

printf**(**"%s-admin$ "**,** username**);**

getInput**(**20**,** command**,** false**);**

sigaction**(**SIGINT**,** **&**act**,** **NULL);**

**if(!**strcmp**(**command**,** "quit"**))** **{**

printf**(**ANSI\_COLOR\_RED"Arrivederci!\n" ANSI\_COLOR\_RESET**);**

**return;**

**}** **else** **if(!**strcmp**(**command**,** "help"**))** **{**

printf**(**ANSI\_COLOR\_CYAN "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" "\n"**);**

printf**(** "\*\*\* Comandi Amministratore \*\*\*" "\n"**);**

printf**(** "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ANSI\_COLOR\_RESET "\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "addlavoratore" ANSI\_COLOR\_RESET " - per aggiungere un lavoratore\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "addazienda" ANSI\_COLOR\_RESET "- per aggiungere un'azienda\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "addpasseggero" ANSI\_COLOR\_RESET "- per aggiungere un passeggero\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "addprenotazione" ANSI\_COLOR\_RESET " - per aggiungere una prenotazione\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "addspedizione" ANSI\_COLOR\_RESET " - per aggiungere una spedizione\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "addservizio" ANSI\_COLOR\_RESET " - per aggiungere un servizio ferroviario\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "addturno" ANSI\_COLOR\_RESET " - per aggiungere un turno\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "serviziotratta" ANSI\_COLOR\_RESET " - per modificare la tratta di un servizio\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "serviziotreno" ANSI\_COLOR\_RESET " - per modificare il treno di un servizio\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "comptreno" ANSI\_COLOR\_RESET " - per ottenere la composizione di un treno\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "configtreno" ANSI\_COLOR\_RESET " - per ottenere la configurazione di un treno\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "assegnaturno" ANSI\_COLOR\_RESET " - per assegnare un turno ad un lavoratore\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "malattia" ANSI\_COLOR\_RESET " - per inserire un lavoratore in malattia\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "freelavoratori" ANSI\_COLOR\_RESET " - per vedere i lavoratori liberi\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "tratte" ANSI\_COLOR\_RESET " - per ottenere le tratte giornaliere\n"**);**

printf**(**ANSI\_COLOR\_MAGENTA "fermatetratta" ANSI\_COLOR\_RESET " - per ottenere le fermate di una tratta\n"**);**

printf**(**ANSI\_COLOR\_YELLOW "quit" ANSI\_COLOR\_RESET " - per uscire dall'applicazione\n"**);**

printf**(**ANSI\_COLOR\_YELLOW "clear" ANSI\_COLOR\_RESET " - per pulire lo schermo\n"**);**

printf**(**ANSI\_COLOR\_CYAN "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ANSI\_COLOR\_RESET "\n"**);**

**}** **else** **if(!**strcmp**(**command**,** "clear"**))** **{**

printf**(**"\033[2J\033[H"**);**

**}** **else** **if(!**strcmp**(**command**,** "addlavoratore"**))** **{**

aggiungi\_lavoratore**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "addturno"**))** **{**

aggiungi\_turno**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "assegnaturno"**))** **{**

assegna\_turno**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "malattia"**))** **{**

richiesta\_malattia**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "addazienda"**))** **{**

aggiungi\_azienda**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "addspedizione"**))** **{**

aggiungi\_spedizione**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "addpasseggero"**))** **{**

aggiungi\_passeggero**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "addprenotazione"**))** **{**

aggiungi\_prenotazione**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "addservizio"**))** **{**

aggiungi\_servizio**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "freelavoratori"**))** **{**

lavoratori\_disponibili**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "serviziotratta"**))** **{**

modifica\_servizio**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "serviziotreno"**))** **{**

modifica\_treno**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "configtreno"**))** **{**

configurazione**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "comptreno"**))** **{**

composizione**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "fermatetratta"**))** **{**

fermate**(**conn**);**

**}** **else** **if(!**strcmp**(**command**,** "tratte"**))** **{**

tratte**(**conn**);**

**}** **else** **{**

printf**(**"comando %s non riconosciuto, digita help per aiuto\n"**,** command**);**

**}**

sigaction**(**SIGINT**,** **&**sa**,** **NULL);**

**}**

**}**

**lavoratore.c**

Utilizzato per l’accesso come lavoratore e per far accedere all’utente a tutte le procedure del suo ruolo.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <signal.h>

#include "defines.h"

#define ANSI\_COLOR\_RED "\x1b[31m"

#define ANSI\_COLOR\_GREEN "\x1b[32m"

#define ANSI\_COLOR\_YELLOW "\x1b[33m"

#define ANSI\_COLOR\_RESET "\x1b[0m"

#define ANSI\_COLOR\_CYAN "\x1b[36m"

static void turni**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**1**];**

char cf**[**16**];**

int status**;**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\nReport Turni\n\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

// Get the required information

printf**(**"CF: "**);**

**if(**getInput**(**45**,** cf**,** false**)** **<** 0**)** **return;**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call report\_lavoratore(?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** cf**;**

param**[**0**].**buffer\_length **=** strlen**(**cf**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'ottenere i turni."**);**

**}**

**do** **{**

dump\_result\_set**(**conn**,** prepared\_stmt**,** "Turni per il lavoratore selezionato"**);**

status **=** mysql\_stmt\_next\_result**(**prepared\_stmt**);**

**if** **(**status **>** 0**)**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Unexpected condition"**,** true**);**

**}** **while** **(**status **==** 0**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

void lavoratore**(**MYSQL **\***conn**,** char **\***username**)** **{**

char command**[**20**];**

printf**(**"\033[2J\033[H"**);**

**if(**mysql\_change\_user**(**conn**,** "lavoratore"**,** "Lineagialla1!"**,**"AziendaFerroviaria"**))** **{**

fprintf**(**stderr**,** "mysql\_change\_user() failed\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

printf**(**ANSI\_COLOR\_GREEN "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

printf**(** "\*Connesso al sistema come lavoratore\*\n"**);**

printf**(** "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" ANSI\_COLOR\_RESET**);**

printf**(**"Inserisci un comando, digita help per aiuto\n\n"**);**

struct sigaction act**,** sa**;**

memset **(&**act**,** '\0'**,** **sizeof(**act**));**

**while(**1**)** **{**

printf**(**"%s-lavoratore$ "**,** username**);**

getInput**(**20**,** command**,** false**);**

sigaction**(**SIGINT**,** **&**act**,** **NULL);**

**if(!**strcmp**(**command**,** "quit"**))** **{**

printf**(**ANSI\_COLOR\_RED"Arrivederci!\n" ANSI\_COLOR\_RESET**);**

**return;**

**}** **else** **if(!**strcmp**(**command**,** "help"**))** **{**

printf**(**ANSI\_COLOR\_GREEN "\*\*\* Comandi lavoratore \*\*\*" ANSI\_COLOR\_RESET "\n"**);**

printf**(**ANSI\_COLOR\_CYAN "turni" ANSI\_COLOR\_RESET " - report dei turni" "\n"**);;**

printf**(**ANSI\_COLOR\_YELLOW "quit" ANSI\_COLOR\_RESET " - per uscire dall'applicazione\n"**);**

printf**(**ANSI\_COLOR\_YELLOW "clear" ANSI\_COLOR\_RESET " - per pulire il terminale\n"**);**

printf**(**ANSI\_COLOR\_GREEN "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ANSI\_COLOR\_RESET "\n"**);**

**}** **else** **if(!**strcmp**(**command**,** "turni"**))** **{**

turni**(**conn**);**

**}** **else** **{**

printf**(**"comando %s non riconosciuto, digita help per aiuto\n"**,** command**);**

**}**

sigaction**(**SIGINT**,** **&**sa**,** **NULL);**

**}**

**}**

**manutentore.c**

Utilizzato per l’accesso come manutentore e per far accedere all’utente a tutte le procedure del suo ruolo.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <signal.h>

#include "defines.h"

#define ANSI\_COLOR\_RED "\x1b[31m"

#define ANSI\_COLOR\_GREEN "\x1b[32m"

#define ANSI\_COLOR\_CYAN "\x1b[36m"

#define ANSI\_COLOR\_RESET "\x1b[0m"

#define ANSI\_COLOR\_YELLOW "\x1b[33m"

static void report**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**4**];**

int id**;**

char info**[**500**];**

int mr\_id**;**

MYSQL\_TIME **\***date**;**

date **=** malloc**(sizeof(**MYSQL\_TIME**));**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nReport Manutenzione\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

// Get the required information

//printf("ID REPORT: ");

//if(getInteger(&id) < 0) return;

printf**(**"Inserisci data di manutenzione(formato dd/mm/yyyy): "**);**

**if(**getDate**(**date**)** **<** 0**)** **return;**

printf**(**"Inserisci ID materiale rotabile: "**);**

**if(**getInteger**(&**mr\_id**)** **<** 0**)** **return;**

printf**(**"Descrizione manutenzione: "**);**

**if(**getInput**(**500**,** info**,** false**)** **<** 0**)** **return;**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call report\_manutenzione(?,?,?,?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**0**].**buffer **=** **&**id**;**

param**[**0**].**buffer\_length **=** **sizeof(**id**);**

param**[**1**].**buffer\_type **=** MYSQL\_TYPE\_DATE**;**

param**[**1**].**buffer **=** date**;**

param**[**1**].**buffer\_length **=** **sizeof(\***date**);**

param**[**2**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**2**].**buffer **=** info**;**

param**[**2**].**buffer\_length **=** strlen**(**info**);**

param**[**3**].**buffer\_type **=** MYSQL\_TYPE\_LONG**;**

param**[**3**].**buffer **=** **&**mr\_id**;**

param**[**3**].**buffer\_length **=** **sizeof(**mr\_id**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nell'inserimento del report."**);**

**goto** out**;**

**}**

printf**(**"Report aggiunto con successo...\n"**);**

out**:**

free**(**date**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

void manutentore**(**MYSQL **\***conn**,** char **\***username**)** **{**

char command**[**20**];**

printf**(**"\033[2J\033[H"**);**

**if(**mysql\_change\_user**(**conn**,** "manutentore"**,** "Lineagialla1!"**,**"AziendaFerroviaria"**))** **{**

fprintf**(**stderr**,** "mysql\_change\_user() failed\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

printf**(**ANSI\_COLOR\_GREEN "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

printf**(** "\*Connesso al sistema come manutentore\*\n"**);**

printf**(** "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" ANSI\_COLOR\_RESET**);**

printf**(**"Inserisci un comando, digita help per aiuto\n\n"**);**

struct sigaction act**,** sa**;**

memset **(&**act**,** '\0'**,** **sizeof(**act**));**

**while(**1**)** **{**

printf**(**"%s-Manutentore$ "**,** username**);**

getInput**(**20**,** command**,** false**);**

sigaction**(**SIGINT**,** **&**act**,** **NULL);**

**if(!**strcmp**(**command**,** "quit"**))** **{**

printf**(**ANSI\_COLOR\_RED"Arrivederci!\n" ANSI\_COLOR\_RESET**);**

**return;**

**}** **else** **if(!**strcmp**(**command**,** "help"**))** **{**

printf**(**ANSI\_COLOR\_GREEN "\*\*\* Comandi manutentore \*\*\*" ANSI\_COLOR\_RESET "\n"**);**

printf**(**ANSI\_COLOR\_CYAN "report" ANSI\_COLOR\_RESET " - report della manutenzione" "\n"**);;**

printf**(**ANSI\_COLOR\_YELLOW "quit" ANSI\_COLOR\_RESET " - per uscire dall'applicazione\n"**);**

printf**(**ANSI\_COLOR\_YELLOW "clear" ANSI\_COLOR\_RESET " - per pulire il terminale\n"**);**

printf**(**ANSI\_COLOR\_GREEN "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ANSI\_COLOR\_RESET "\n"**);**

**}** **else** **if(!**strcmp**(**command**,** "report"**))** **{**

report**(**conn**);**

**}** **else** **{**

printf**(**"comando %s non riconosciuto, digita help per aiuto\n"**,** command**);**

**}**

sigaction**(**SIGINT**,** **&**sa**,** **NULL);**

**}**

**}**

**controllore.c**

Utilizzato per l’accesso come controllore e per far accedere all’utente a tutte le procedure del suo ruolo.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <signal.h>

#include "defines.h"

#define ANSI\_COLOR\_RED "\x1b[31m"

#define ANSI\_COLOR\_GREEN "\x1b[32m"

#define ANSI\_COLOR\_RESET "\x1b[0m"

#define ANSI\_COLOR\_CYAN "\x1b[36m"

#define ANSI\_COLOR\_YELLOW "\x1b[33m"

static void verifica**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**1**];**

char codice**[**5**];**

int status**;**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nVerifica Prenotazione\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

// Get the required information

printf**(**"Codice Prenotazione: "**);**

**if(**getInput**(**5**,** codice**,** false**)** **<** 0**)** **return;**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call verifica\_prenotazione(?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** codice**;**

param**[**0**].**buffer\_length **=** strlen**(**codice**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nella verifica della prenotazione."**);**

**}**

**do** **{**

dump\_result\_set**(**conn**,** prepared\_stmt**,** ""**);**

status **=** mysql\_stmt\_next\_result**(**prepared\_stmt**);**

**if** **(**status **>** 0**)**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Unexpected condition"**,** true**);**

**}** **while** **(**status **==** 0**);**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

static void convalida**(**MYSQL **\***conn**)** **{**

MYSQL\_STMT **\***prepared\_stmt**;**

MYSQL\_BIND param**[**1**];**

char codice**[**5**];**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nConvalida Prenotazione\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

// Get the required information

printf**(**"Codice Prenotazione: "**);**

**if(**getInput**(**5**,** codice**,** false**)** **<** 0**)** **return;**

// Prepare stored procedure call

**if(!**setup\_prepared\_stmt**(&**prepared\_stmt**,** "call convalida\_prenotazione(?)"**,** conn**))** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Impossibile inizializzare lo statement.\n"**,** false**);**

**}**

// Prepare parameters

memset**(**param**,** 0**,** **sizeof(**param**));**

param**[**0**].**buffer\_type **=** MYSQL\_TYPE\_VAR\_STRING**;**

param**[**0**].**buffer **=** codice**;**

param**[**0**].**buffer\_length **=** strlen**(**codice**);**

**if** **(**mysql\_stmt\_bind\_param**(**prepared\_stmt**,** param**)** **!=** 0**)** **{**

finish\_with\_stmt\_error**(**conn**,** prepared\_stmt**,** "Errore nel binding dei parametri.\n"**,** true**);**

**}**

// Run procedure

**if** **(**mysql\_stmt\_execute**(**prepared\_stmt**)** **!=** 0**)** **{**

print\_stmt\_error **(**prepared\_stmt**,** "Errore nella convalida della prenotazione."**);**

**}else** **{**

printf**(**"Convalida eseguita correttamente\n"**);**

**}**

mysql\_stmt\_close**(**prepared\_stmt**);**

**}**

void controllore**(**MYSQL **\***conn**,** char **\***username**)** **{**

char command**[**20**];**

printf**(**"\033[2J\033[H"**);**

**if(**mysql\_change\_user**(**conn**,** "controllore"**,** "Lineagialla1!"**,**"AziendaFerroviaria"**))** **{**

fprintf**(**stderr**,** "mysql\_change\_user() failed\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

printf**(**ANSI\_COLOR\_GREEN "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

printf**(** "\*Connesso al sistema come controllore\*\n"**);**

printf**(** "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" ANSI\_COLOR\_RESET**);**

printf**(**"Inserisci un comando, digita help per aiuto\n\n"**);**

struct sigaction act**,** sa**;**

memset **(&**act**,** '\0'**,** **sizeof(**act**));**

**while(**1**)** **{**

printf**(**"%s-Controllore$ "**,** username**);**

getInput**(**20**,** command**,** false**);**

sigaction**(**SIGINT**,** **&**act**,** **NULL);**

**if(!**strcmp**(**command**,** "quit"**))** **{**

printf**(**ANSI\_COLOR\_RED"Arrivederci!\n" ANSI\_COLOR\_RESET**);**

**return;**

**}** **else** **if(!**strcmp**(**command**,** "help"**))** **{**

printf**(**ANSI\_COLOR\_GREEN "\*\*\*\*\* Comandi controllore \*\*\*\*\*" ANSI\_COLOR\_RESET "\n"**);**

printf**(**ANSI\_COLOR\_CYAN "verifica" ANSI\_COLOR\_RESET " - verifica della prenotazione" "\n"**);;**

printf**(**ANSI\_COLOR\_CYAN "convalida" ANSI\_COLOR\_RESET " - convalida prenotazione" "\n"**);;**

printf**(**ANSI\_COLOR\_YELLOW "quit" ANSI\_COLOR\_RESET " - per uscire dall'applicazione\n"**);**

printf**(**ANSI\_COLOR\_YELLOW "clear" ANSI\_COLOR\_RESET " - per pulire il terminale\n"**);**

printf**(**ANSI\_COLOR\_GREEN "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" ANSI\_COLOR\_RESET "\n"**);**

**}** **else** **if(!**strcmp**(**command**,** "verifica"**))** **{**

verifica**(**conn**);**

**}else** **if(!**strcmp**(**command**,** "convalida"**))** **{**

convalida**(**conn**);**

**}** **else** **{**

printf**(**"comando %s non riconosciuto, digita help per aiuto\n"**,** command**);**

**}**

sigaction**(**SIGINT**,** **&**sa**,** **NULL);**

**}**

**}**

**utils.c**

Contiene tutte le funzioni utili per interagire con la base di dati nel codice c.

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <signal.h>

#include <termios.h>

#include <errno.h>

#include "defines.h"

static volatile sig\_atomic\_t signo**;**

**typedef** struct sigaction sigaction\_t**;**

static void handler**(**int s**);**

void print\_stmt\_error **(**MYSQL\_STMT **\***stmt**,** char **\***message**)**

**{**

fprintf **(**stderr**,** "%s\n"**,** message**);**

**if** **(**stmt **!=** **NULL)** **{**

fprintf **(**stderr**,** "Error %u (%s): %s\n"**,**

mysql\_stmt\_errno **(**stmt**),**

mysql\_stmt\_sqlstate**(**stmt**),**

mysql\_stmt\_error **(**stmt**));**

**}**

**}**

void print\_error**(**MYSQL **\***conn**,** char **\***message**)**

**{**

fprintf **(**stderr**,** "%s\n"**,** message**);**

**if** **(**conn **!=** **NULL)** **{**

#if MYSQL\_VERSION\_ID >= 40101

fprintf **(**stderr**,** "Error %u (%s): %s\n"**,**

mysql\_errno **(**conn**),** mysql\_sqlstate**(**conn**),** mysql\_error **(**conn**));**

#else

fprintf **(**stderr**,** "Error %u: %s\n"**,**

mysql\_errno **(**conn**),** mysql\_error **(**conn**));**

#endif

**}**

**}**

void finish\_with\_error**(**MYSQL **\***conn**,** char **\***message**)**

**{**

print\_error**(**conn**,** message**);**

mysql\_close**(**conn**);**

exit**(**EXIT\_FAILURE**);**

**}**

void finish\_with\_stmt\_error**(**MYSQL **\***conn**,** MYSQL\_STMT **\***stmt**,** char **\***message**,** bool close\_stmt**)**

**{**

print\_stmt\_error**(**stmt**,** message**);**

**if(**close\_stmt**)** mysql\_stmt\_close**(**stmt**);**

mysql\_close**(**conn**);**

exit**(**EXIT\_FAILURE**);**

**}**

bool setup\_prepared\_stmt**(**MYSQL\_STMT **\*\***stmt**,** char **\***statement**,** MYSQL **\***conn**)**

**{**

bool update\_length **=** true**;**

**\***stmt **=** mysql\_stmt\_init**(**conn**);**

**if** **(\***stmt **==** **NULL)**

**{**

print\_error**(**conn**,** "Could not initialize statement handler"**);**

**return** false**;**

**}**

**if** **(**mysql\_stmt\_prepare **(\***stmt**,** statement**,** strlen**(**statement**))** **!=** 0**)** **{**

print\_stmt\_error**(\***stmt**,** "Could not prepare statement"**);**

**return** false**;**

**}**

mysql\_stmt\_attr\_set**(\***stmt**,** STMT\_ATTR\_UPDATE\_MAX\_LENGTH**,** **&**update\_length**);**

**return** true**;**

**}**

//TODO refactor from this

int getInput**(**unsigned int lung**,** char **\***stringa**,** bool hide**)** **{**

char c**;**

unsigned int i**;**

// Dichiara le variabili necessarie ad un possibile mascheramento dell'input

struct sigaction sa**,** savealrm**,** saveint**,** savehup**,** savequit**,** saveterm**;**

struct sigaction savetstp**,** savettin**,** savettou**;**

struct termios term**,** oterm**;**

**if(**hide**)** **{**

// Svuota il buffer

**(**void**)** fflush**(**stdout**);**

// Cattura i segnali che altrimenti potrebbero far terminare il programma, lasciando l'utente senza output sulla shell

sigemptyset**(&**sa**.**sa\_mask**);**

sa**.**sa\_flags **=** SA\_INTERRUPT**;** // Per non resettare le system call

sa**.**sa\_handler **=** handler**;**

**(**void**)** sigaction**(**SIGALRM**,** **&**sa**,** **&**savealrm**);**

**(**void**)** sigaction**(**SIGINT**,** **&**sa**,** **&**saveint**);**

**(**void**)** sigaction**(**SIGHUP**,** **&**sa**,** **&**savehup**);**

**(**void**)** sigaction**(**SIGQUIT**,** **&**sa**,** **&**savequit**);**

**(**void**)** sigaction**(**SIGTERM**,** **&**sa**,** **&**saveterm**);**

**(**void**)** sigaction**(**SIGTSTP**,** **&**sa**,** **&**savetstp**);**

**(**void**)** sigaction**(**SIGTTIN**,** **&**sa**,** **&**savettin**);**

**(**void**)** sigaction**(**SIGTTOU**,** **&**sa**,** **&**savettou**);**

// Disattiva l'output su schermo

**if** **(**tcgetattr**(**fileno**(**stdin**),** **&**oterm**)** **==** 0**)** **{**

**(**void**)** memcpy**(&**term**,** **&**oterm**,** **sizeof(**struct termios**));**

term**.**c\_lflag **&=** **~(**ECHO**|**ECHONL**);**

**(**void**)** tcsetattr**(**fileno**(**stdin**),** TCSAFLUSH**,** **&**term**);**

**}** **else** **{**

**(**void**)** memset**(&**term**,** 0**,** **sizeof(**struct termios**));**

**(**void**)** memset**(&**oterm**,** 0**,** **sizeof(**struct termios**));**

**}**

**}**

// Acquisisce da tastiera al pi� lung - 1 caratteri

**for(**i **=** 0**;** i **<** lung**;** i**++)** **{**

int size **=** fread**(&**c**,** **sizeof(**char**),** 1**,** stdin**);**

**if(**size **==** 0 **&&** errno **==** EINTR**)** **{**

**if(**hide**)** **{**

**(**void**)** tcsetattr**(**fileno**(**stdin**),** TCSAFLUSH**,** **&**oterm**);**

**(**void**)** sigaction**(**SIGALRM**,** **&**savealrm**,** **NULL);**

**(**void**)** sigaction**(**SIGINT**,** **&**saveint**,** **NULL);**

**(**void**)** sigaction**(**SIGHUP**,** **&**savehup**,** **NULL);**

**(**void**)** sigaction**(**SIGQUIT**,** **&**savequit**,** **NULL);**

**(**void**)** sigaction**(**SIGTERM**,** **&**saveterm**,** **NULL);**

**(**void**)** sigaction**(**SIGTSTP**,** **&**savetstp**,** **NULL);**

**(**void**)** sigaction**(**SIGTTIN**,** **&**savettin**,** **NULL);**

**(**void**)** sigaction**(**SIGTTOU**,** **&**savettou**,** **NULL);**

**}**

**return** **-**1**;**

**}**

**if(**c **==** '\n'**)** **{**

stringa**[**i**]** **=** '\0'**;**

**break;**

**}** **else**

stringa**[**i**]** **=** c**;**

// Gestisce gli asterischi

**if(**hide**)** **{**

**if(**c **==** '\b'**)** // Backspace

**(**void**)** write**(**fileno**(**stdout**),** **&**c**,** **sizeof(**char**));**

**else**

**(**void**)** write**(**fileno**(**stdout**),** "\*"**,** **sizeof(**char**));**

**}**

**}**

// Controlla che il terminatore di stringa sia stato inserito

**if(**i **==** lung **-** 1**)**

stringa**[**i**]** **=** '\0'**;**

// Se sono stati digitati pi� caratteri, svuota il buffer della tastiera

**if(**strlen**(**stringa**)** **>=** lung**)** **{**

// Svuota il buffer della tastiera

**do** **{**

c **=** getchar**();**

**}** **while** **(**c **!=** '\n'**);**

**}**

**if(**hide**)** **{**

//L'a capo dopo l'input

**(**void**)** write**(**fileno**(**stdout**),** "\n"**,** 1**);**

// Ripristina le impostazioni precedenti dello schermo

**(**void**)** tcsetattr**(**fileno**(**stdin**),** TCSAFLUSH**,** **&**oterm**);**

// Ripristina la gestione dei segnali

**(**void**)** sigaction**(**SIGALRM**,** **&**savealrm**,** **NULL);**

**(**void**)** sigaction**(**SIGINT**,** **&**saveint**,** **NULL);**

**(**void**)** sigaction**(**SIGHUP**,** **&**savehup**,** **NULL);**

**(**void**)** sigaction**(**SIGQUIT**,** **&**savequit**,** **NULL);**

**(**void**)** sigaction**(**SIGTERM**,** **&**saveterm**,** **NULL);**

**(**void**)** sigaction**(**SIGTSTP**,** **&**savetstp**,** **NULL);**

**(**void**)** sigaction**(**SIGTTIN**,** **&**savettin**,** **NULL);**

**(**void**)** sigaction**(**SIGTTOU**,** **&**savettou**,** **NULL);**

// Se era stato ricevuto un segnale viene rilanciato al processo stesso

**if(**signo**)**

**(**void**)** raise**(**signo**);**

**}**

**return** strlen**(**stringa**);**

**}**

// Per la gestione dei segnali

static void handler**(**int s**)** **{**

signo **=** s**;**

**}**

static void print\_dashes**(**MYSQL\_RES **\***res\_set**)**

**{**

MYSQL\_FIELD **\***field**;**

unsigned int i**,** j**;**

mysql\_field\_seek**(**res\_set**,** 0**);**

putchar**(**'+'**);**

**for** **(**i **=** 0**;** i **<** mysql\_num\_fields**(**res\_set**);** i**++)** **{**

field **=** mysql\_fetch\_field**(**res\_set**);**

**for** **(**j **=** 0**;** j **<** field**->**max\_length **+** 2**;** j**++)**

putchar**(**'-'**);**

putchar**(**'+'**);**

**}**

putchar**(**'\n'**);**

**}**

static void dump\_result\_set\_header**(**MYSQL\_RES **\***res\_set**)**

**{**

MYSQL\_FIELD **\***field**;**

unsigned long col\_len**;**

unsigned int i**;**

/\* determine column display widths -- requires result set to be \*/

/\* generated with mysql\_store\_result(), not mysql\_use\_result() \*/

mysql\_field\_seek **(**res\_set**,** 0**);**

**for** **(**i **=** 0**;** i **<** mysql\_num\_fields **(**res\_set**);** i**++)** **{**

field **=** mysql\_fetch\_field **(**res\_set**);**

col\_len **=** strlen**(**field**->**name**);**

**if** **(**col\_len **<** field**->**max\_length**)**

col\_len **=** field**->**max\_length**;**

**if** **(**col\_len **<** 4 **&&** **!**IS\_NOT\_NULL**(**field**->**flags**))**

col\_len **=** 4**;** /\* 4 = length of the word "NULL" \*/

field**->**max\_length **=** col\_len**;** /\* reset column info \*/

**}**

print\_dashes**(**res\_set**);**

putchar**(**'|'**);**

mysql\_field\_seek **(**res\_set**,** 0**);**

**for** **(**i **=** 0**;** i **<** mysql\_num\_fields**(**res\_set**);** i**++)** **{**

field **=** mysql\_fetch\_field**(**res\_set**);**

printf**(**" %-\*s |"**,** **(**int**)**field**->**max\_length**,** field**->**name**);**

**}**

putchar**(**'\n'**);**

print\_dashes**(**res\_set**);**

**}**

void dump\_result\_set**(**MYSQL **\***conn**,** MYSQL\_STMT **\***stmt**,** char **\***title**)**

**{**

int i**;**

int status**;**

int num\_fields**;** /\* number of columns in result \*/

MYSQL\_FIELD **\***fields**;** /\* for result set metadata \*/

MYSQL\_BIND **\***rs\_bind**;** /\* for output buffers \*/

MYSQL\_RES **\***rs\_metadata**;**

MYSQL\_TIME **\***date**;**

MYSQL\_TIME **\***time**;**

size\_t attr\_size**;**

/\* Prefetch the whole result set. This in conjunction with

\* STMT\_ATTR\_UPDATE\_MAX\_LENGTH set in `setup\_prepared\_stmt`

\* updates the result set metadata which are fetched in this

\* function, to allow to compute the actual max length of

\* the columns.

\*/

**if** **(**mysql\_stmt\_store\_result**(**stmt**))** **{**

fprintf**(**stderr**,** " mysql\_stmt\_execute(), 1 failed\n"**);**

fprintf**(**stderr**,** " %s\n"**,** mysql\_stmt\_error**(**stmt**));**

exit**(**0**);**

**}**

/\* the column count is > 0 if there is a result set \*/

/\* 0 if the result is only the final status packet \*/

num\_fields **=** mysql\_stmt\_field\_count**(**stmt**);**

**if** **(**num\_fields **>** 0**)** **{**

/\* there is a result set to fetch \*/

printf**(**"%s\n"**,** title**);**

**if((**rs\_metadata **=** mysql\_stmt\_result\_metadata**(**stmt**))** **==** **NULL)** **{**

finish\_with\_stmt\_error**(**conn**,** stmt**,** "Unable to retrieve result metadata\n"**,** true**);**

**}**

dump\_result\_set\_header**(**rs\_metadata**);**

fields **=** mysql\_fetch\_fields**(**rs\_metadata**);**

rs\_bind **=** **(**MYSQL\_BIND **\*)**malloc**(sizeof** **(**MYSQL\_BIND**)** **\*** num\_fields**);**

**if** **(!**rs\_bind**)** **{**

finish\_with\_stmt\_error**(**conn**,** stmt**,** "Cannot allocate output buffers\n"**,** true**);**

**}**

memset**(**rs\_bind**,** 0**,** **sizeof** **(**MYSQL\_BIND**)** **\*** num\_fields**);**

/\* set up and bind result set output buffers \*/

**for** **(**i **=** 0**;** i **<** num\_fields**;** **++**i**)** **{**

// Properly size the parameter buffer

**switch(**fields**[**i**].**type**)** **{**

**case** MYSQL\_TYPE\_DATE**:**

**case** MYSQL\_TYPE\_TIMESTAMP**:**

**case** MYSQL\_TYPE\_DATETIME**:**

**case** MYSQL\_TYPE\_TIME**:**

attr\_size **=** **sizeof(**MYSQL\_TIME**);**

**break;**

**case** MYSQL\_TYPE\_FLOAT**:**

attr\_size **=** **sizeof(**float**);**

**break;**

**case** MYSQL\_TYPE\_DOUBLE**:**

attr\_size **=** **sizeof(**double**);**

**break;**

**case** MYSQL\_TYPE\_TINY**:**

attr\_size **=** **sizeof(**signed char**);**

**break;**

**case** MYSQL\_TYPE\_SHORT**:**

**case** MYSQL\_TYPE\_YEAR**:**

attr\_size **=** **sizeof(**short int**);**

**break;**

**case** MYSQL\_TYPE\_LONG**:**

**case** MYSQL\_TYPE\_INT24**:**

attr\_size **=** **sizeof(**int**);**

**break;**

**case** MYSQL\_TYPE\_LONGLONG**:**

attr\_size **=** **sizeof(**int**);**

**break;**

**default:**

attr\_size **=** fields**[**i**].**max\_length**;**

**break;**

**}**

// Setup the binding for the current parameter

rs\_bind**[**i**].**buffer\_type **=** fields**[**i**].**type**;**

rs\_bind**[**i**].**buffer **=** malloc**(**attr\_size **+** 1**);**

rs\_bind**[**i**].**buffer\_length **=** attr\_size **+** 1**;**

**if(**rs\_bind**[**i**].**buffer **==** **NULL)** **{**

finish\_with\_stmt\_error**(**conn**,** stmt**,** "Cannot allocate output buffers\n"**,** true**);**

**}**

**}**

**if(**mysql\_stmt\_bind\_result**(**stmt**,** rs\_bind**))** **{**

finish\_with\_stmt\_error**(**conn**,** stmt**,** "Unable to bind output parameters\n"**,** true**);**

**}**

/\* fetch and display result set rows \*/

**while** **(**true**)** **{**

status **=** mysql\_stmt\_fetch**(**stmt**);**

**if** **(**status **==** 1 **||** status **==** MYSQL\_NO\_DATA**)**

**break;**

putchar**(**'|'**);**

**for** **(**i **=** 0**;** i **<** num\_fields**;** i**++)** **{**

**if** **(**rs\_bind**[**i**].**is\_null\_value**)** **{**

printf **(**" %-\*s |"**,** **(**int**)**fields**[**i**].**max\_length**,** "NULL"**);**

**continue;**

**}**

**switch** **(**rs\_bind**[**i**].**buffer\_type**)** **{**

**case** MYSQL\_TYPE\_VAR\_STRING**:**

**case** MYSQL\_TYPE\_DATETIME**:**

printf**(**" %-\*s |"**,** **(**int**)**fields**[**i**].**max\_length**,** **(**char**\*)**rs\_bind**[**i**].**buffer**);**

**break;**

**case** MYSQL\_TYPE\_DATE**:**

**case** MYSQL\_TYPE\_TIMESTAMP**:**

date **=** **(**MYSQL\_TIME **\*)**rs\_bind**[**i**].**buffer**;**

printf**(**" %d-%02d-%02d |"**,** date**->**year**,** date**->**month**,** date**->**day**);**

**break;**

**case** MYSQL\_TYPE\_STRING**:**

printf**(**" %-\*s |"**,** **(**int**)**fields**[**i**].**max\_length**,** **(**char **\*)**rs\_bind**[**i**].**buffer**);**

**break;**

**case** MYSQL\_TYPE\_FLOAT**:**

**case** MYSQL\_TYPE\_DOUBLE**:**

printf**(**" %.02f |"**,** **\*(**float **\*)**rs\_bind**[**i**].**buffer**);**

**break;**

**case** MYSQL\_TYPE\_LONG**:**

**case** MYSQL\_TYPE\_SHORT**:**

**case** MYSQL\_TYPE\_TINY**:**

**case** MYSQL\_TYPE\_LONGLONG**:**

printf**(**" %-\*d |"**,** **(**int**)**fields**[**i**].**max\_length**,** **\*(**int **\*)**rs\_bind**[**i**].**buffer**);**

**break;**

**case** MYSQL\_TYPE\_NEWDECIMAL**:**

printf**(**" %-\*.02lf |"**,** **(**int**)**fields**[**i**].**max\_length**,** **\*(**float**\*)** rs\_bind**[**i**].**buffer**);**

**break;**

**case** MYSQL\_TYPE\_TIME**:**

time **=** **(**MYSQL\_TIME **\*)**rs\_bind**[**i**].**buffer**;**

printf**(**"%.02d:%.02d %\*s|"**,** time**->**hour**,** time**->**minute**,** **(**int**)**fields**[**i**].**max\_length **-** **(**int**)**strlen**(**"hh:mm"**),** ""**);**

**break;**

**default:**

printf**(**"ERROR: Unhandled type (%d)\n"**,** rs\_bind**[**i**].**buffer\_type**);**

abort**();**

**}**

**}**

putchar**(**'\n'**);**

print\_dashes**(**rs\_metadata**);**

**}**

mysql\_free\_result**(**rs\_metadata**);** /\* free metadata \*/

/\* free output buffers \*/

**for** **(**i **=** 0**;** i **<** num\_fields**;** i**++)** **{**

free**(**rs\_bind**[**i**].**buffer**);**

**}**

free**(**rs\_bind**);**

**}**

**}**

int getInteger**(**int **\***dest**)** **{**

int size **=** scanf**(**"%d"**,** dest**);**

**if(**size **<** 0 **&&** errno **==** EINTR**)** **{**

**return** **-**1**;**

**}**

getchar**();**

**return** size**;**

**}**

int getDate**(**MYSQL\_TIME **\***date**)** **{**

unsigned int day**,** month**,** year**;**

int size **=** scanf**(**"%d/%d/%d"**,&**day**,&**month**,&**year**);**

**if(**size **<** 0 **&&** errno **==** EINTR**)** **{**

**return** **-**1**;**

**}**

getchar**();**

date**->**day **=** day**;**

date**->**month **=** month**;**

date**->**year **=** year**;**

**return** size**;**

**}**

int getTime**(**MYSQL\_TIME **\***time**)** **{**

int size **=** scanf**(**"%u:%u"**,&**time**->**hour**,** **&**time**->**minute**);**

**if(**size **<** 0 **&&** errno **==** EINTR**)** **{**

**return** **-**1**;**

**}**

time**->**second **=** 0**;**

**return** size**;**

**}**

**defines.h**

#include <stdbool.h>

#include <mysql.h>

extern void print\_stmt\_error **(**MYSQL\_STMT **\***stmt**,** char **\***message**);**

extern bool setup\_prepared\_stmt**(**MYSQL\_STMT **\*\***stmt**,** char **\***statement**,** MYSQL **\***conn**);**

extern void print\_error **(**MYSQL **\***conn**,** char **\***message**);**

int getInput**(**unsigned int lung**,** char **\***stringa**,** bool hide**);**

extern void lavoratore**(**MYSQL **\***conn**,** char **\***username**);**

extern void controllore**(**MYSQL **\***conn**,** char **\***username**);**

extern void manutentore**(**MYSQL **\***conn**,** char **\***username**);**

extern void admin**(**MYSQL **\***conn**,** char **\***username**);**

extern void dump\_result\_set**(**MYSQL **\***conn**,** MYSQL\_STMT **\***stmt**,** char **\***title**);**

extern void finish\_with\_error**(**MYSQL **\***conn**,** char **\***message**);**

extern void finish\_with\_stmt\_error**(**MYSQL **\***conn**,** MYSQL\_STMT **\***stmt**,** char **\***message**,** bool close\_stmt**);**

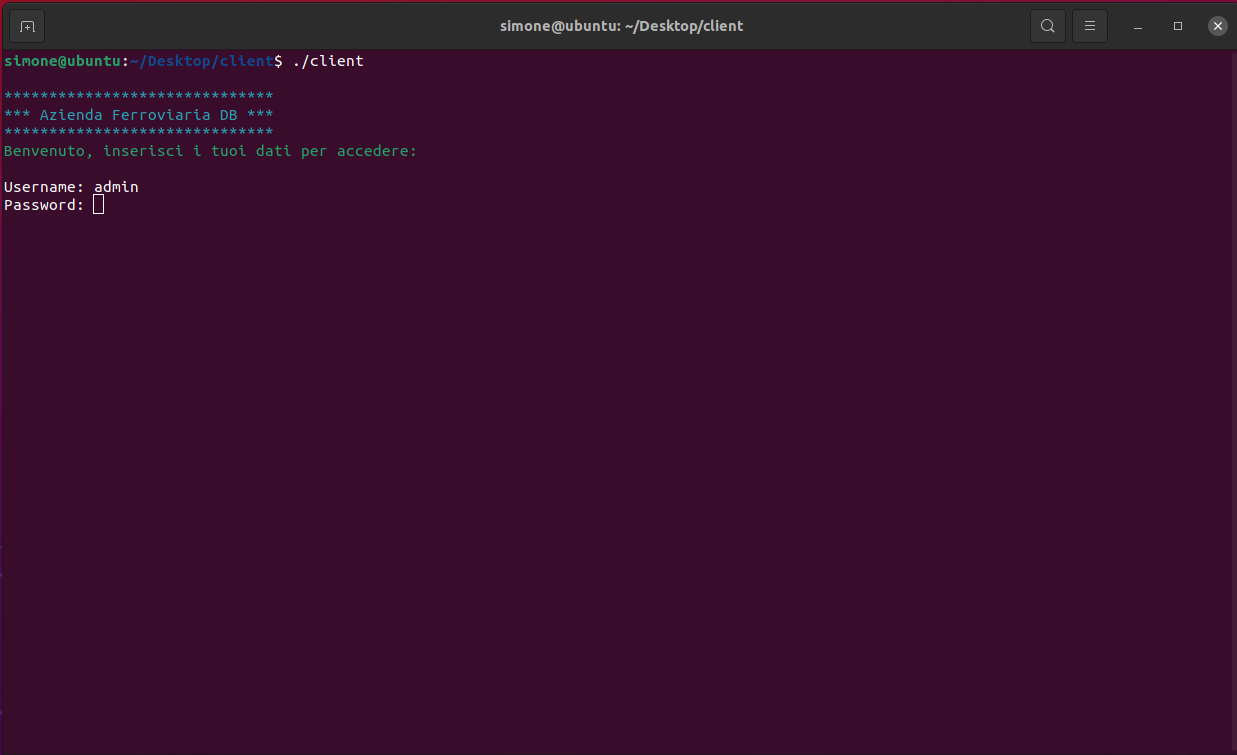
extern void dump\_result\_set**(**MYSQL **\***conn**,** MYSQL\_STMT **\***stmt**,** char **\***title**);**

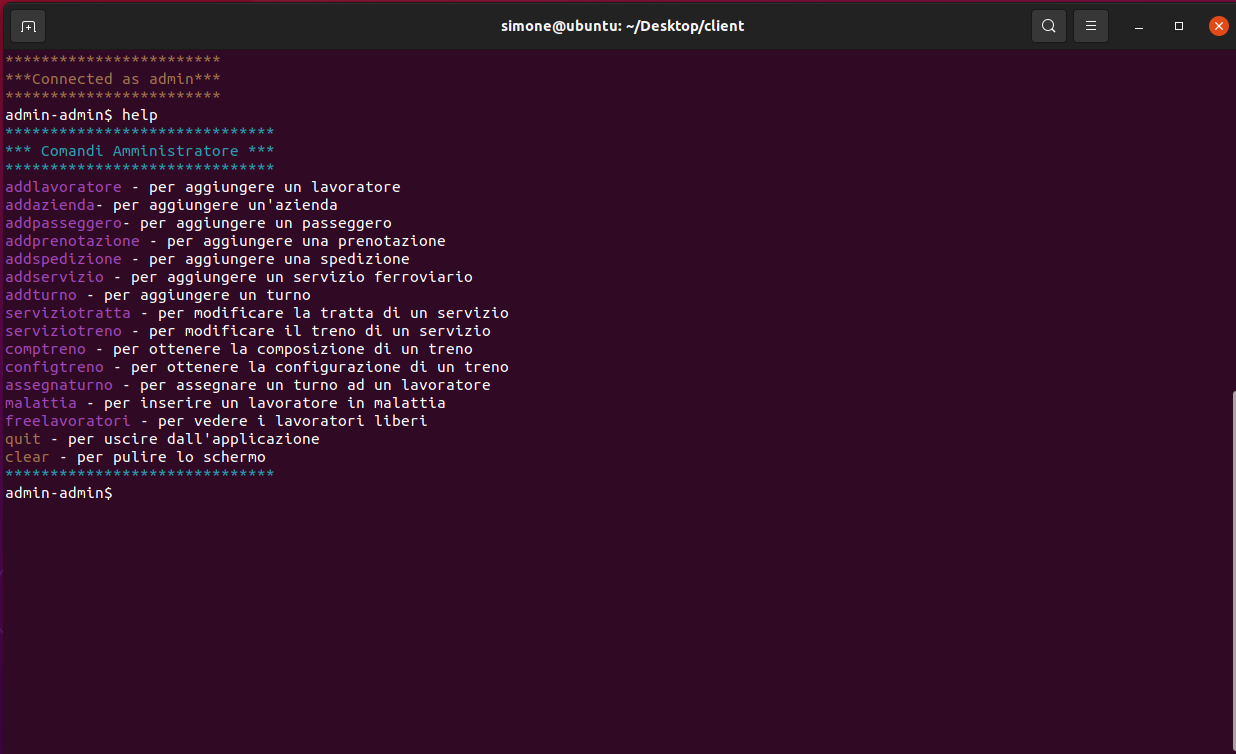
int getInteger**(**int **\***dest**);**

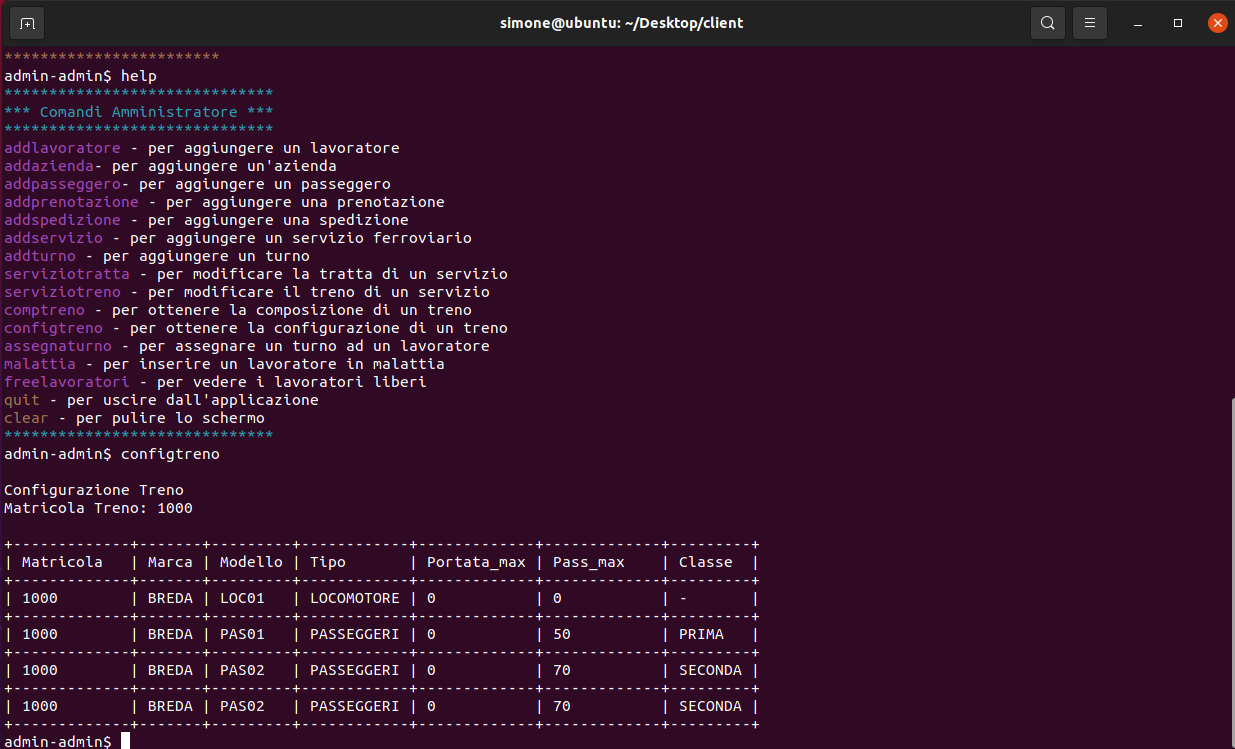
int getDate**(**MYSQL\_TIME **\***date**);**

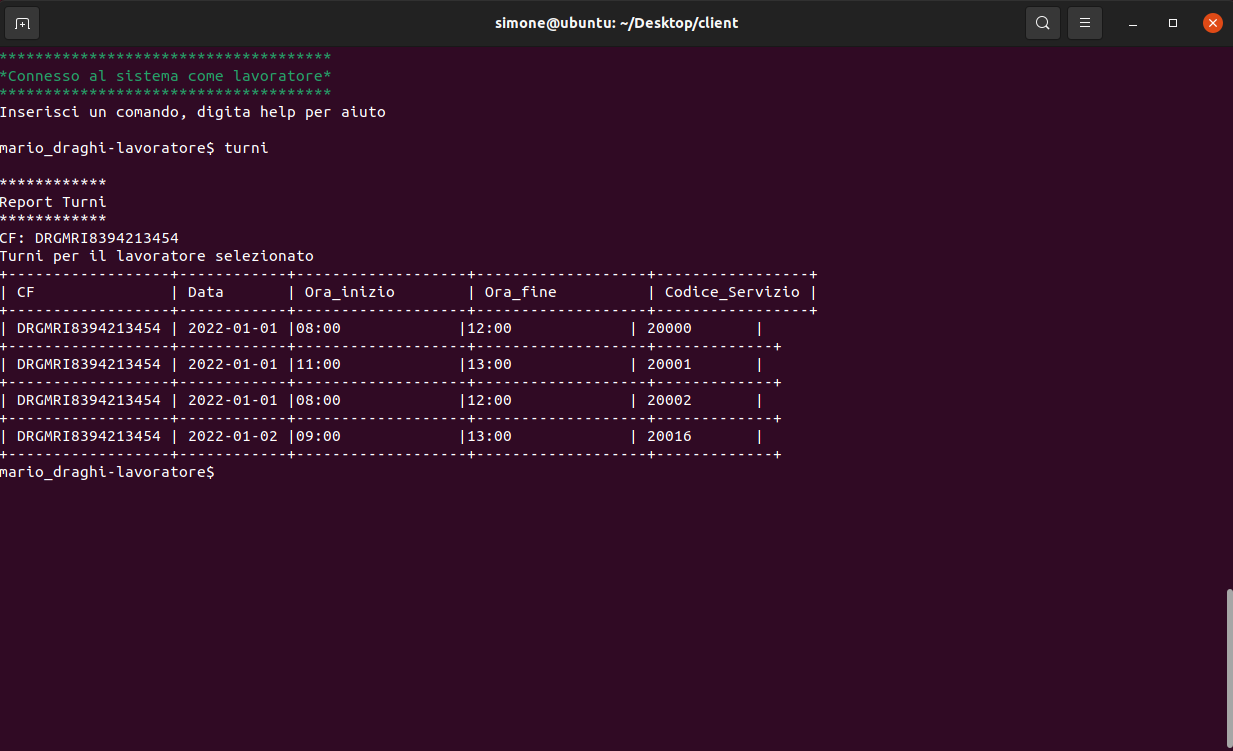
int getTime**(**MYSQL\_TIME **\***time**);**

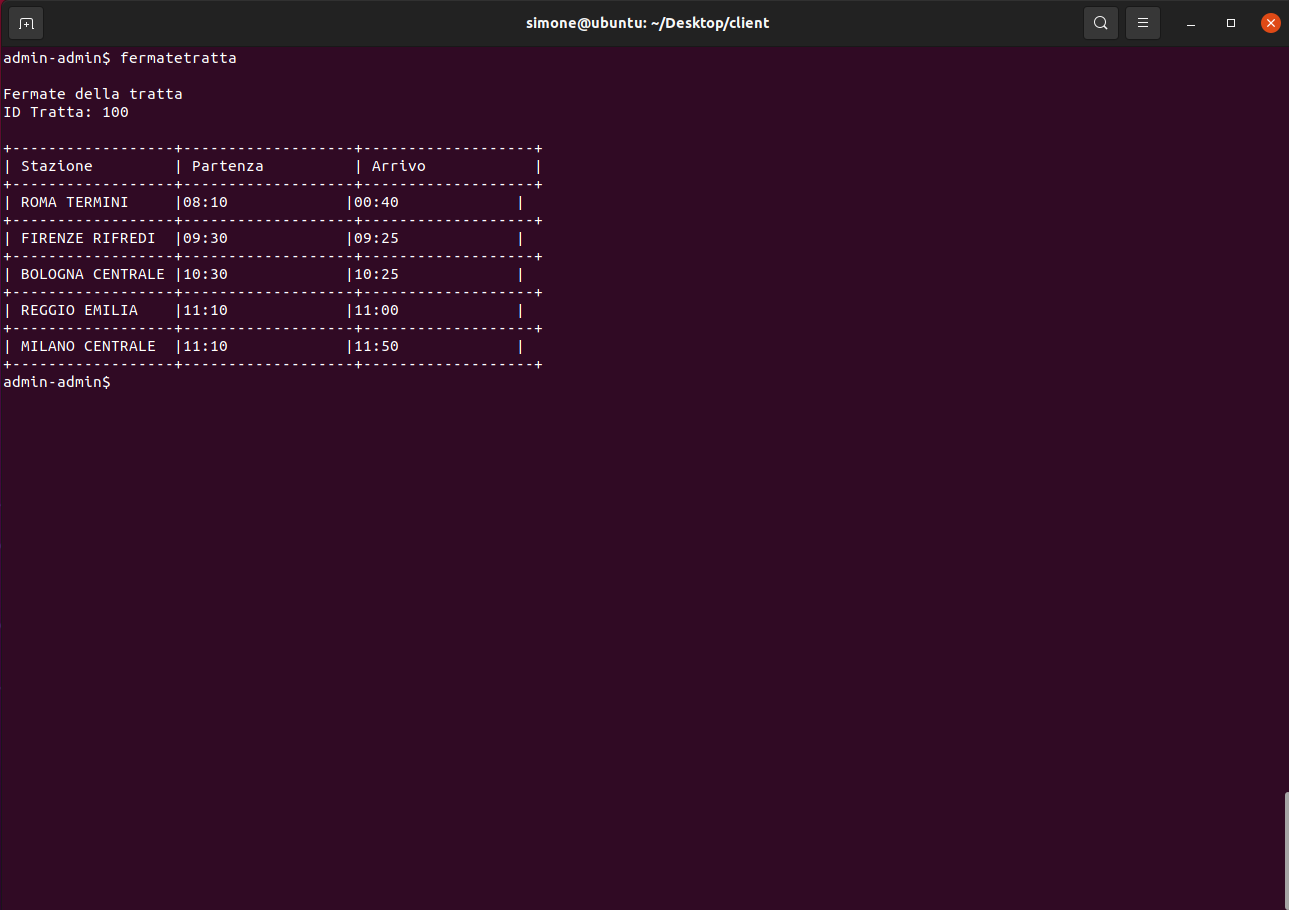
## Print screen del Front End

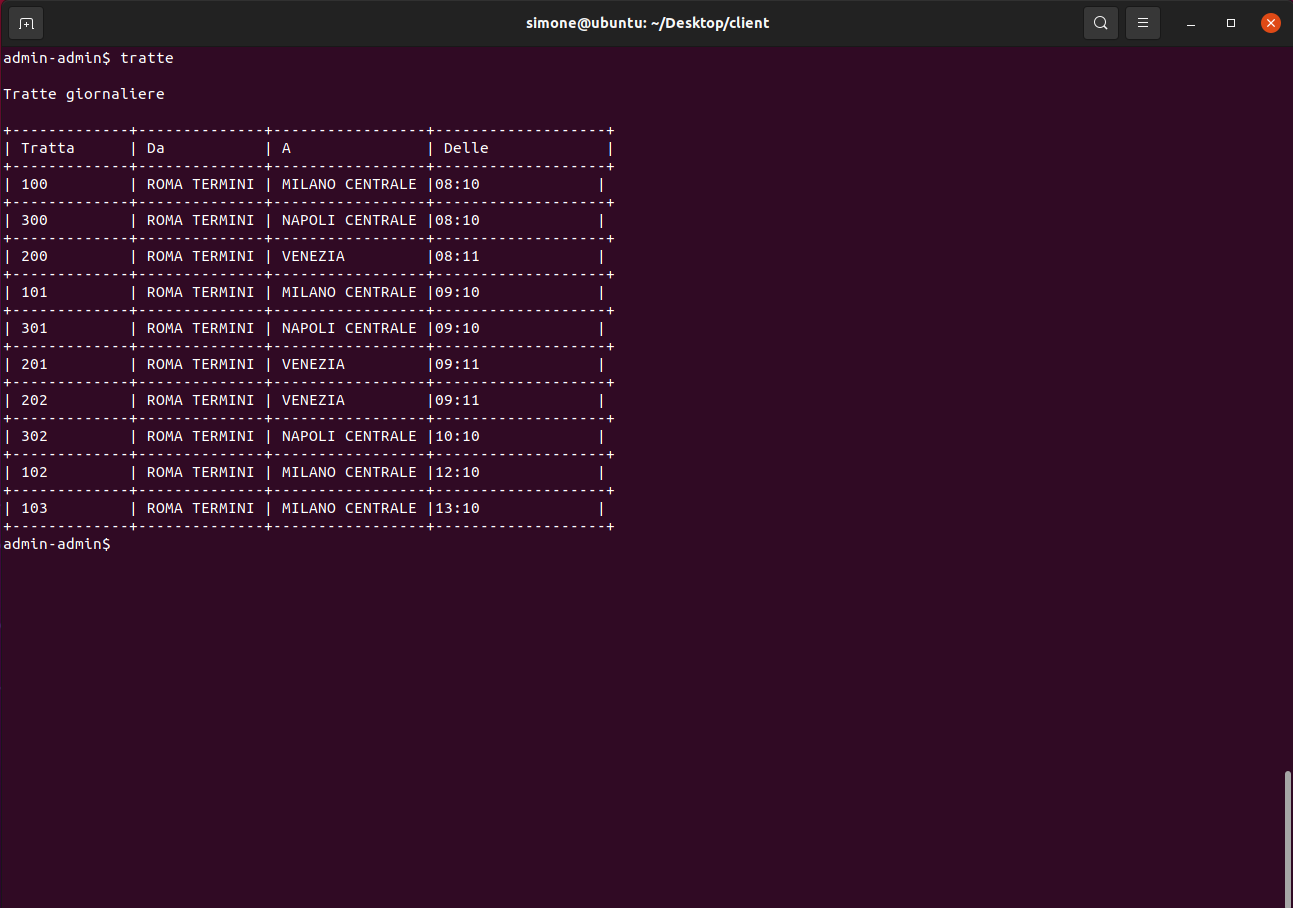


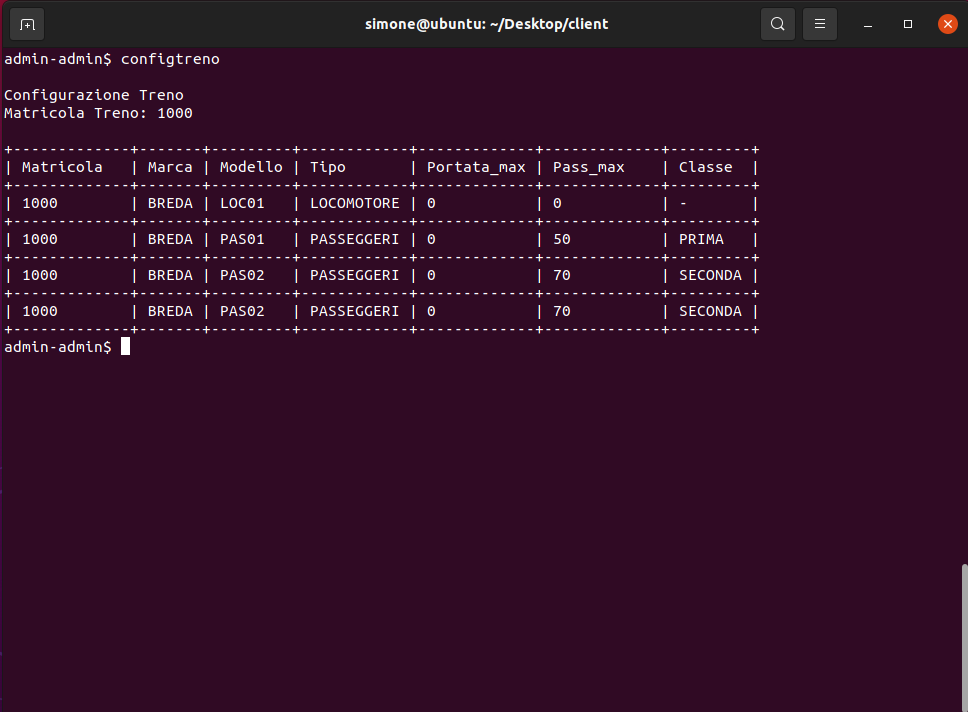












1. Indicare con E le entità, con R le relazioni [↑](#footnote-ref-1)
2. PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna. [↑](#footnote-ref-2)
3. IDX = index, UQ = unique, FT = full text, PR = primary. [↑](#footnote-ref-3)